UNIVERSITÀ DI PISA

Scuola di Ingegneria

Corso di Laurea in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Analisi CFD della resistenza aerodinamica di un velivolo

Relatori:

Prof. Ing. Giovanni Lombardi

Ing. Marco Maganzi

Ing. Salati Loris

Candidato:

Sebastiano Pinto

Anno Accademico 2014-2015

Indice

INDICE DELLE FIGURE

INDICE DELLE TABELLE

Sommario

Lo scopo di questa tesi, realizzata in collaborazione con OMASUD, è quello di analizzare la distribuzione delle forze aerodinamiche di resistenza agenti sul velivolo tramite simulazioni CFD. Una volta trovate le zone, per così dire, "critiche" si cercherà di proporre soluzioni alternative per ridurre la resistenza complessiva del velivolo. Al fine di ottimizzare la riduzione della resistenza per un'intera missione di volo si è analizzata la fase di crociera.

La tesi è stata sviluppata partendo dalla preparazione del modello e della mesh di superficie con il software BETA CAE ANSA v 15 per poi utilizzare il software STAR CCM+ per la mesh di volume e la simulazione.

In seguito sono state eseguite alcune modifiche alla geometria del velivolo e si è fatta un'analisi dei risultati fra le varie configurazioni.

1 Introduzione

1.1 Generalità

Figura 1.1: Skycar

Lo SkyCar è un velivolo compatto bimotore ad ala alta con motori a elica spingente. Ha una forma del tutto non convenzionale, basta guardare la fusoliera: presenta una configurazione a doppia trave di coda e il piano di coda orizzontale è completamente mobile.

La fusoliera è stata progettata in modo tale da permettere la presenza di ampie finestrature e, nella sua configurazione base, ha 5 posti suddivisi su 2 file: la fila anteriore a due postazioni per il pilota ed eventuale co-pilota, la posteriore per tre passeggeri.

L'ala, di pianta rettangolare e con le winglet all'estremità, è montata alta e presenta le due gondole motore, dove sono alloggiati i propulsori, in posizione arretrata rispetto al bordo d'attacco dell'ala.

Il carrello d'atterraggio presenta una configurazione triciclo anteriore, retraibile fin dentro gli appositi alloggi (posti all'esterno della fusoliera). E' comandato tramite attuatori elettrici con possibilità di comando meccanico in caso di emergenza.

La propulsione è affidata a due 2 motori Lycoming IO-360-C1E6 montati in configurazione spingente. Gli IO-360, motori con configurazione a 4 cilindri contrapposti, raffreddati ad aria e dotati di iniezione elettronica, sono capaci di erogare una potenza pari a 200 hp (149 kW) ciascuno e nello SkyCar sono abbinati ad eliche Hartzell bipala a passo variabile a controllo idraulico.

Nella Tabella 1 sono riportate le principali caratteristiche del velivolo:

DIMENSIONI	
Lunghezza	8,82 m
Altezza	2,70 m
Apertura alare	12 m
PESO	
MTOW	1995 kg
SISTEMA PROPULSIVO	
Motore	2 Lycoming IO-360-C1E6
Potenza massima	200 hp ciascuno
Elica	2-Blade Hartzell

Tabella 1: Caratteristiche tecniche Skycar

Figura 1.2: Interno Skycar

Come già anticipato nel sommario, il velivolo è stato analizzato nella configurazione di crociera alla velocità di 140 kt (pari a circa 260 km/h). Le equazioni di bilancio che governano il velivolo in volo livellato sono:

$$\begin{cases} P_d = P_n \\ L = W \end{cases}$$

Dove con P_d e P_n si sono indicati rispettivamente potenza disponibile e potenza necessaria. Indicando con η_e il rendimento dell'elica e con P la potenza erogata dal motore, si avrà:

$$P_d = \eta_e P$$

Il termine di potenza necessaria può invece essere espresso nella forma:

$$P_n = DV$$

Da queste semplici relazioni è possibile ricavare la velocità di crociera data la potenza disponibile e la resistenza complessiva del velivolo.

Figura 1.3: Skycar in volo

1.2 Obiettivi

L'obiettivo della presente tesi è quello di valutare inizialmente le forze aerodinamiche agenti sul velivolo soffermandosi in particolar modo su quelle di resistenza. Dopodiché si cercherà di distribuire tali forza sulle componenti del velivolo per capire quali sono quelli che danno un contributo maggiore alla resistenza.

Infine si proporranno delle modifiche geometriche allo scopo di ridurre la resistenza e verranno fatte delle analisi CFD apposite per valutarne l'effettivo andamento.

2 Modello geometrico

2.1 Importazione modello

Il modello CAD del velivolo *Skycar* è stato fornito dalla OMA SUD Aircrafts in formato igs. Per lo scopo della tesi si è ritenuto opportuno studiare una geometria semplificata del velivolo, senza particolari zone ad elevato dettaglio e senza parti interne al velivolo. Per l'importazione della geometria CAD nel software non è stata necessaria alcuna conversione del formato file perché l'estensione ".igs" è compatibile con BETA CAE ANSA v15. In Figura 2.1 viene riportata l'immagine del modello appena importato:

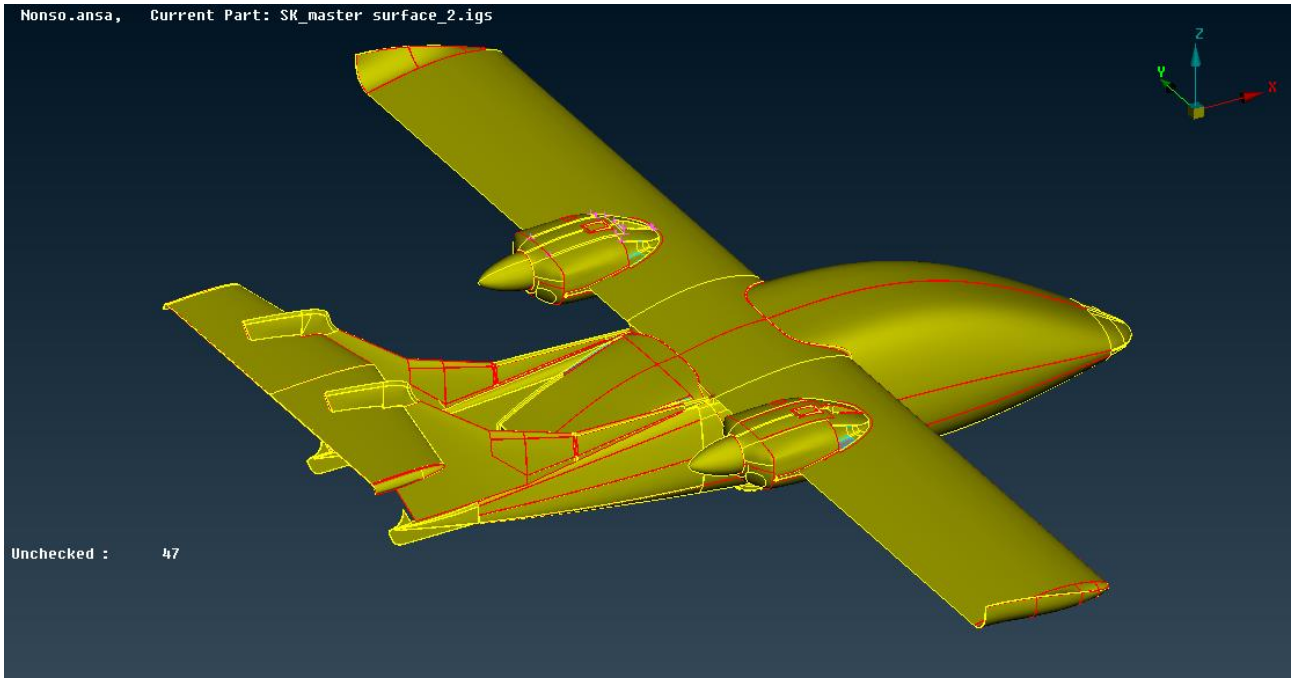


Figura 2.1: Velivolo importato in BETA CAE ANSA v15

2.2 Geometry clean up

Secondo la convenzione rappresentativa di ANSA, le linee rosse stanno ad indicare che esse sono connesse ad un'unica superficie, le linee gialle sono connesse a 2 superfici, le linee blu sono connesse a 3 o più superfici. Risulta abbastanza intuitivo che il modello richiede una fase di pulizia della geometria. Il risultato finale dovrà essere un unico volume chiuso con elementi tutti connessi fra loro e rappresentanti solo l'esterno del velivolo. Come risulterà più chiaro nel capitolo dedicato alla mesh, vanno eliminati anche tutti quegli elementi interni al velivolo che comportano la presenza delle linee blu.

Il primo comando usato in ANSA per inizializzare la pulizia è FACES -> TOPO che corregge automaticamente le incongruenze più grossolane delle connessioni. Si riporta in Figura 2.2 la geometria a seguito del comando TOPO:

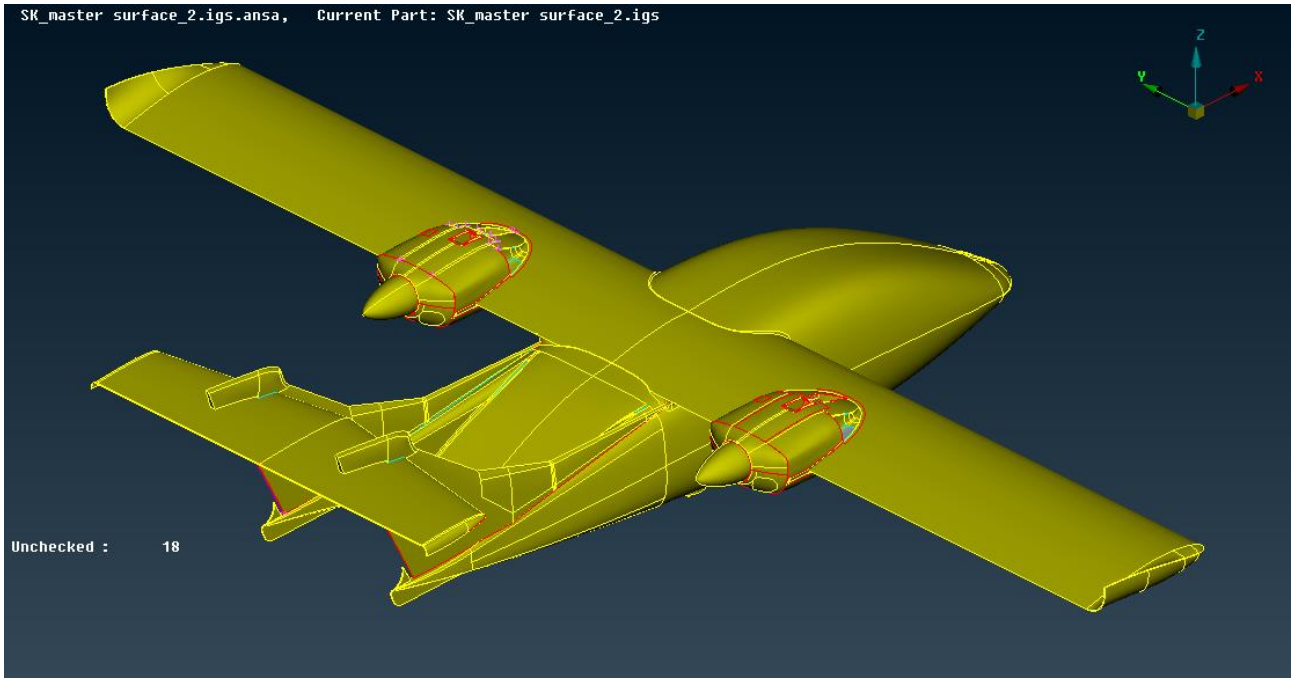
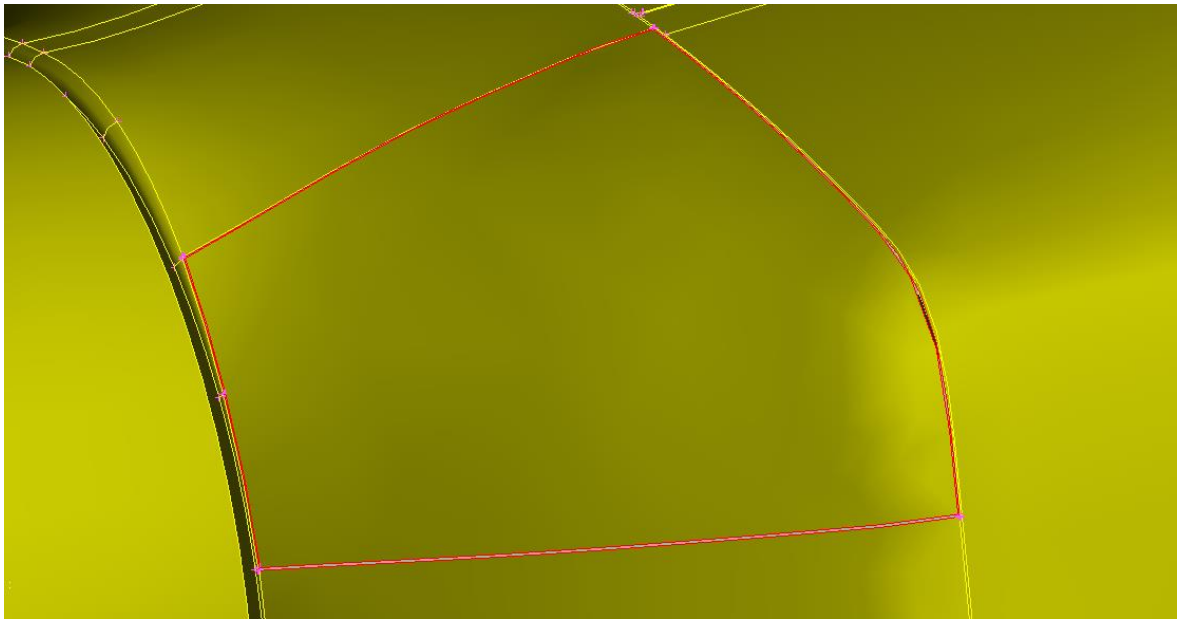
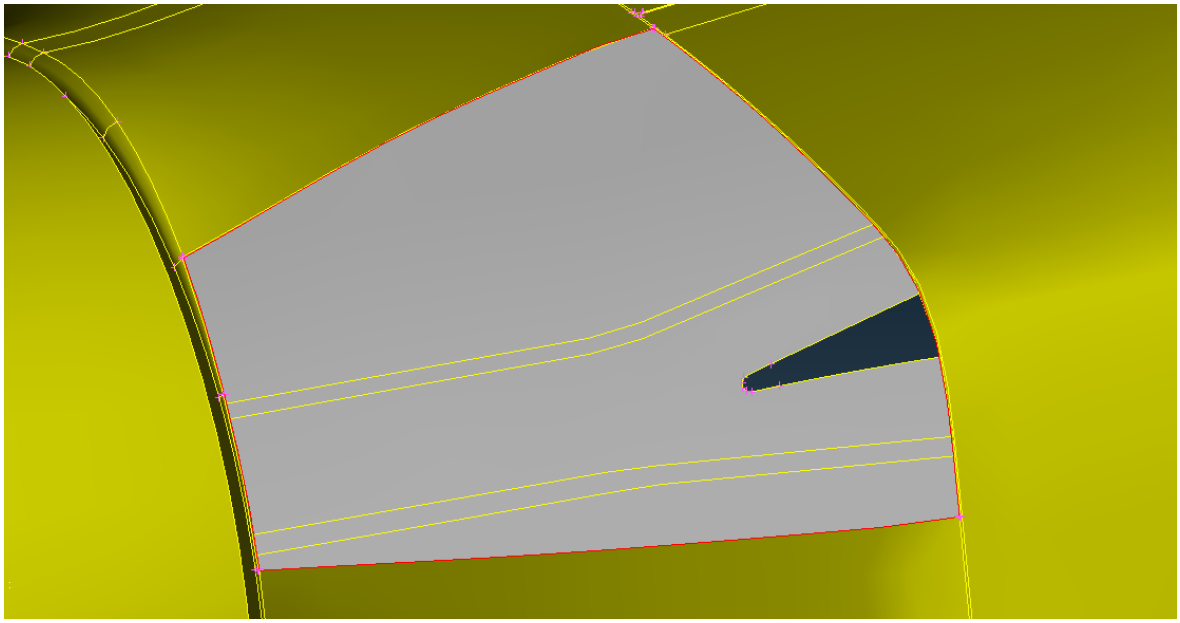


Figura 2.2: Prima pulizia con il comando TOPO

Successivamente si è provveduto a correggere manualmente le restanti connessioni. L'idea base è quella di aggiungere delle superfici fra gli spazi delimitati dalle linee rosse. Il comando utilizzato per lo scopo è FACE -> CREATE -> COONS. In accordo con la convenzione di creazione di superfici implementata in ANSA, è opportuno selezionare i COONS che

delimitano la superficie che vogliamo aggiungere uno di seguito all'altro mantenendo lo stesso senso (orario o antiorario che sia). A volte, quando la geometria delle superfici da creare risulta troppo complessa, è stato necessario cancellare suddette superfici e rifarle. Il comando utilizzato per la cancellazione è FACE -> DELETE. Nelle figure seguenti vediamo un esempio della procedura di pulizia della geometria nella gondola motore:





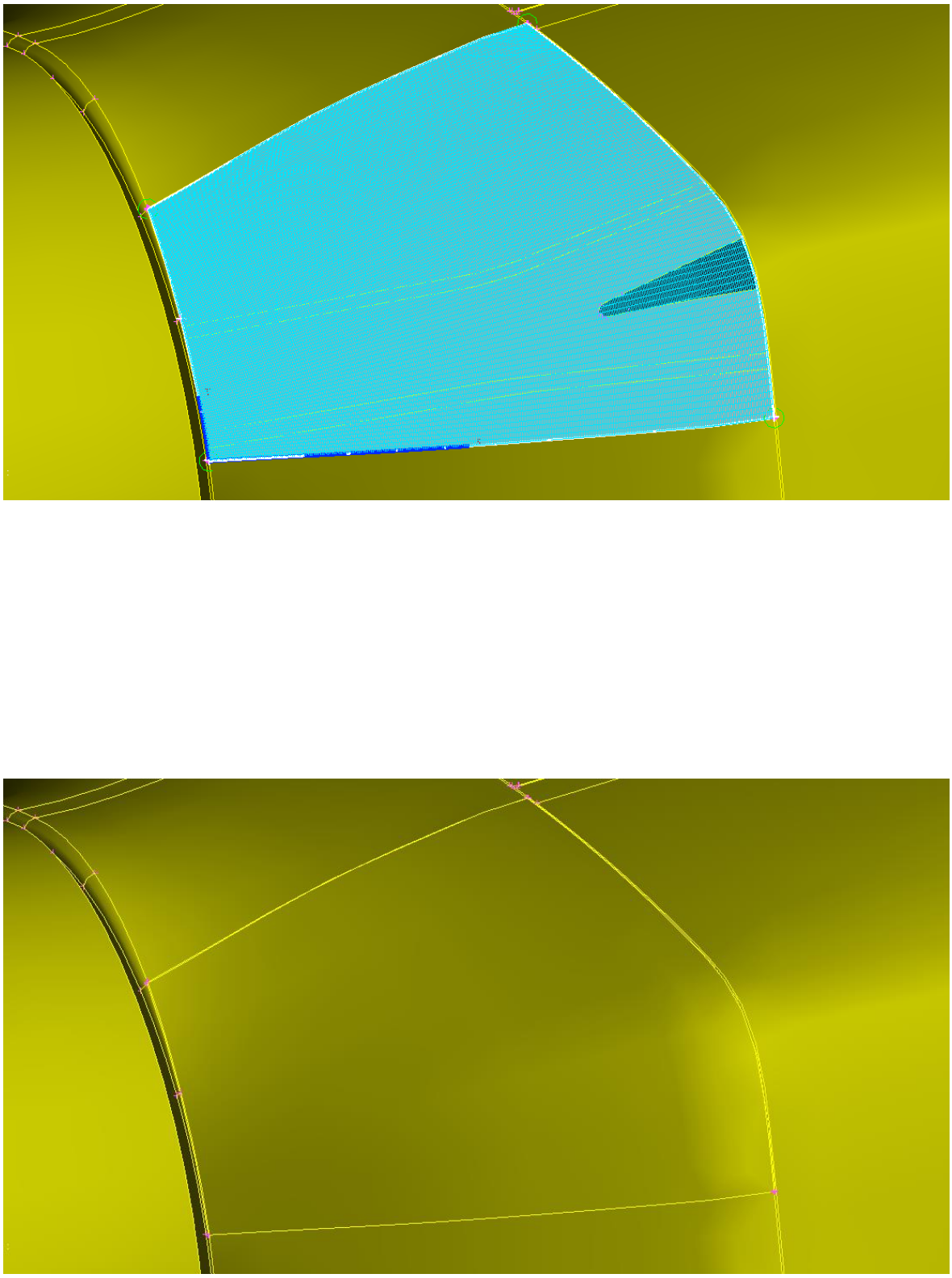


Figura 2.3: Pulizia della geometria

Alla fine del procedimento, il modello ottenuto è rappresentato in Figura 2.4:

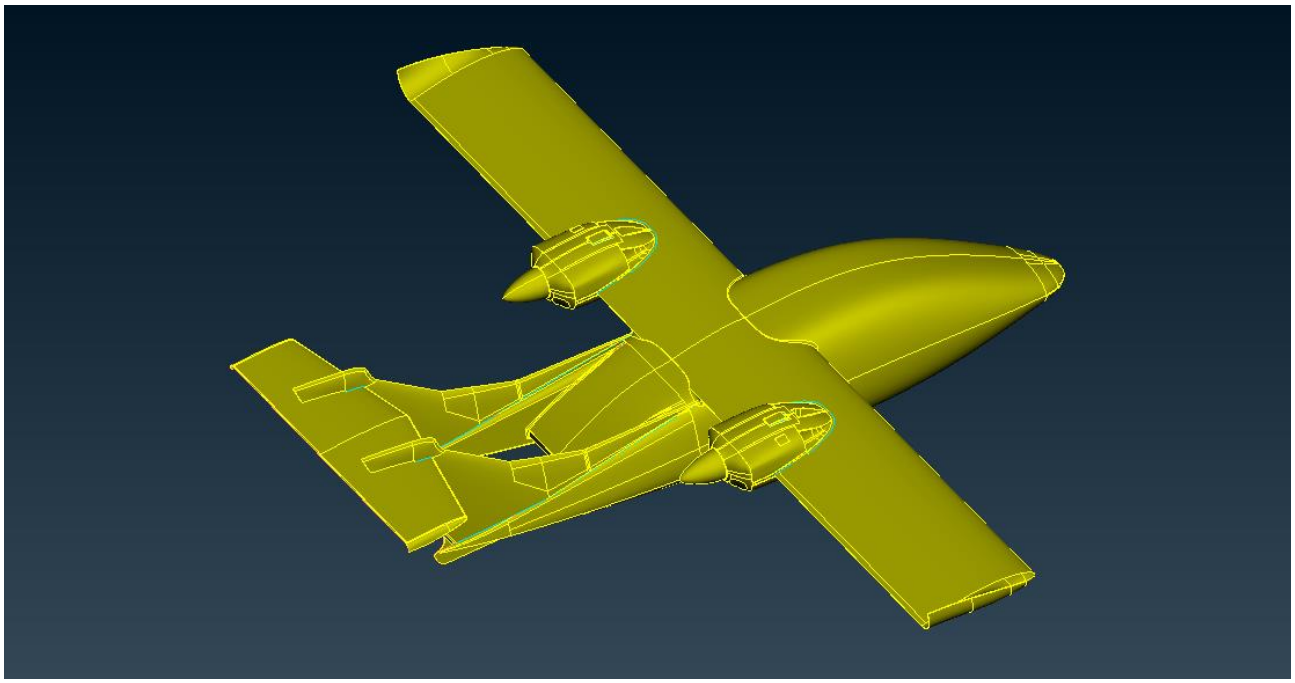
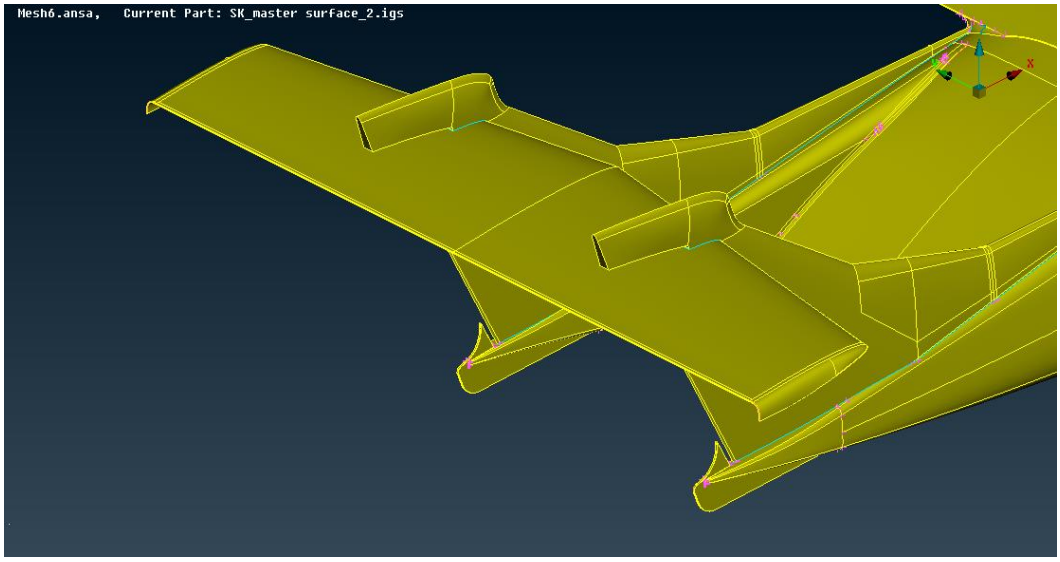


Figura 2.4: Eliminazione linee rosse

Trattandosi di correzioni fatte a mano, può sempre capitare che qualche errore sfugga. Proprio per questo vengono utilizzati i comandi CHECK -> GEOMETRY e CHECK -> INTERSECTION. Il software è in grado di correggere automaticamente tutte le tipologie di errore presenti in CHECK -> GEOMETRY, tranne i TRIPLE CONS, ovvero le linee blu che risultano connesse a 3 o più superfici. In questo caso occorre eliminare manualmente le superfici in eccesso. Un esempio di questo procedimento è rappresentato nelle Figura 2.5:



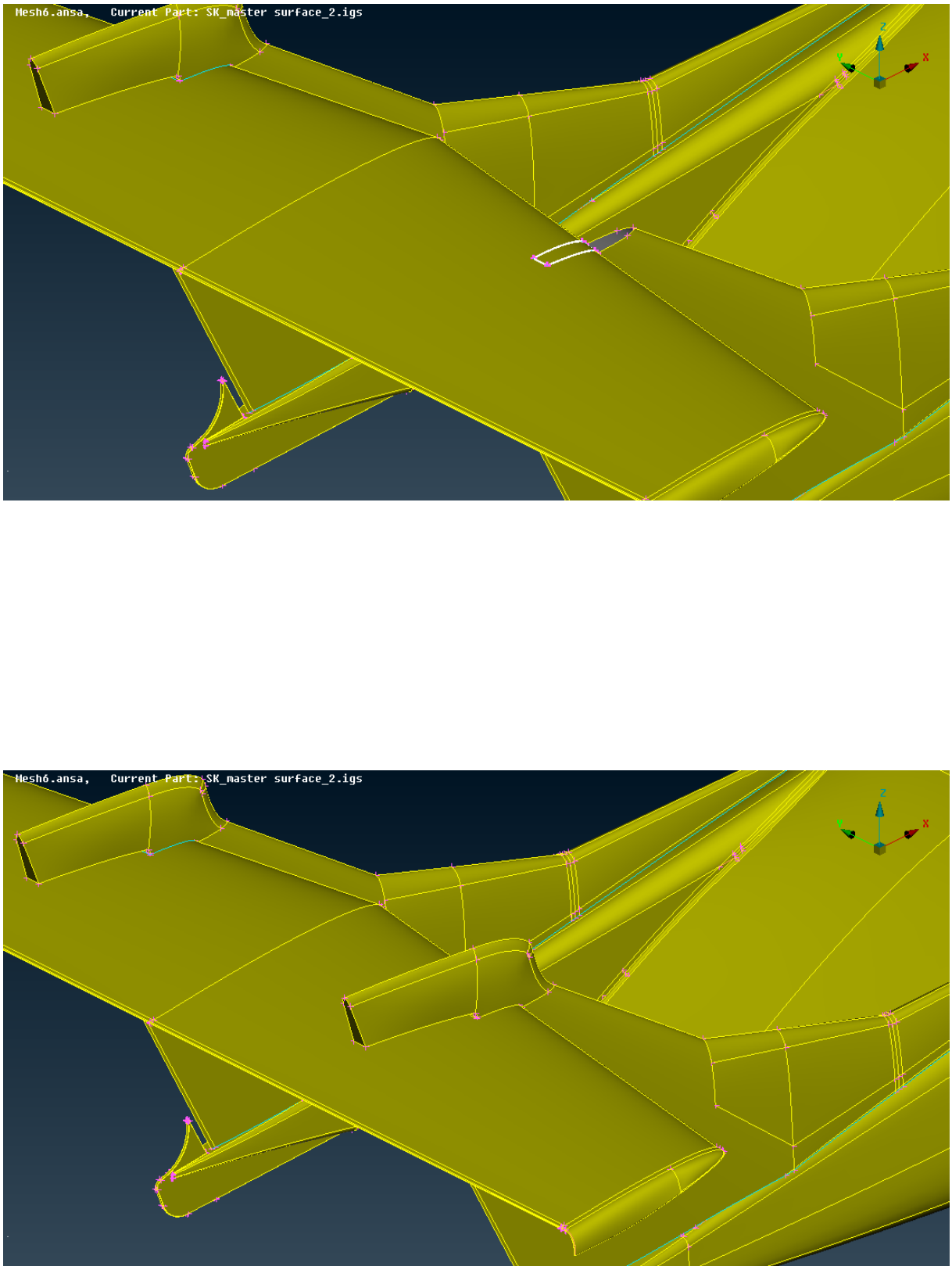


Figura 2.5: Correzione linee blu

Infine, in Figura 2.6**,** viene mostrato il velivolo a conclusione della pulizia della geometria. L'assenza di linee rosse o blu indica una corretta connessione fra tutti gli elementi superficiali.

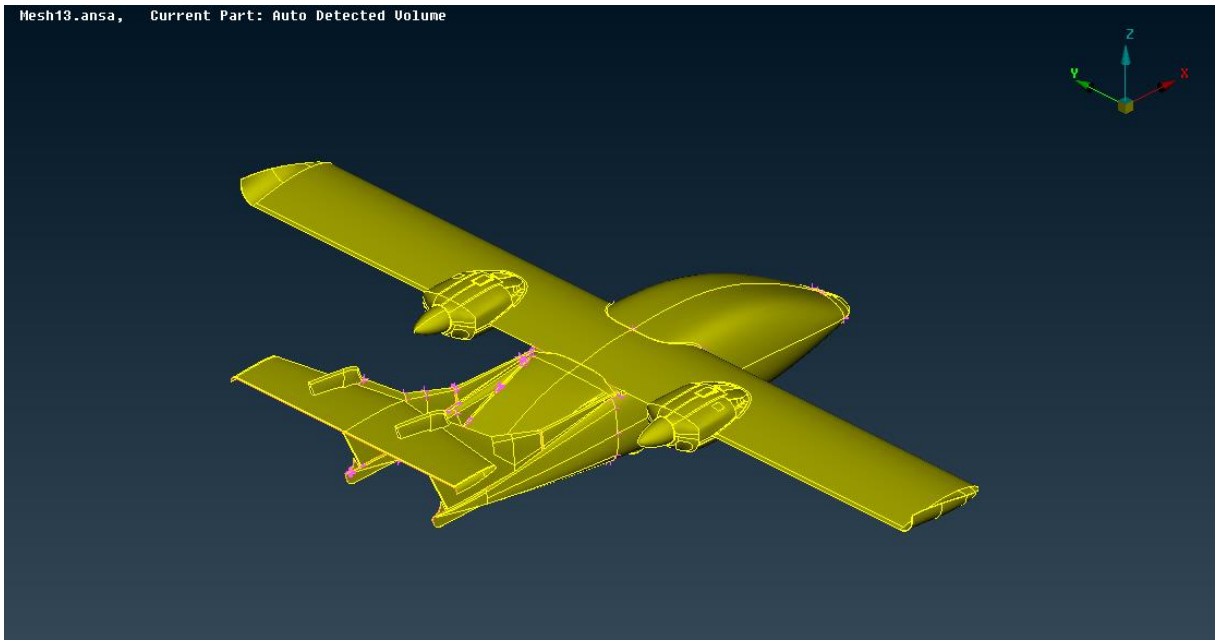


Figura 2.6: Fase di Geometry Clean Up completata

2.3 Volume di controllo

L'ultimo passo, prima di iniziare la mesh che vedremo nel capitolo successivo, è la definizione di un box nel quale racchiudere il modello, che determina il limite del dominio di calcolo. Le dimensioni del volume di controllo vanno opportunamente scelte, poiché un dominio troppo grande non è necessario ai fini della simulazione e inoltre richiederebbe un enorme costo computazionale; invece un dominio troppo piccolo potrebbe invalidare l'accuratezza dei risultati.

Il dominio scelto è:

- 18*L di lunghezza di cui 5*L a monte del velivolo e 12*L a valle;
- 8*b di larghezza;
- 6*b di altezza;

dove con L si è indicato la lunghezza del velivolo e con b l'apertura alare.

Si riporta in Figura 2.7 l'immagine del dominio così definito:

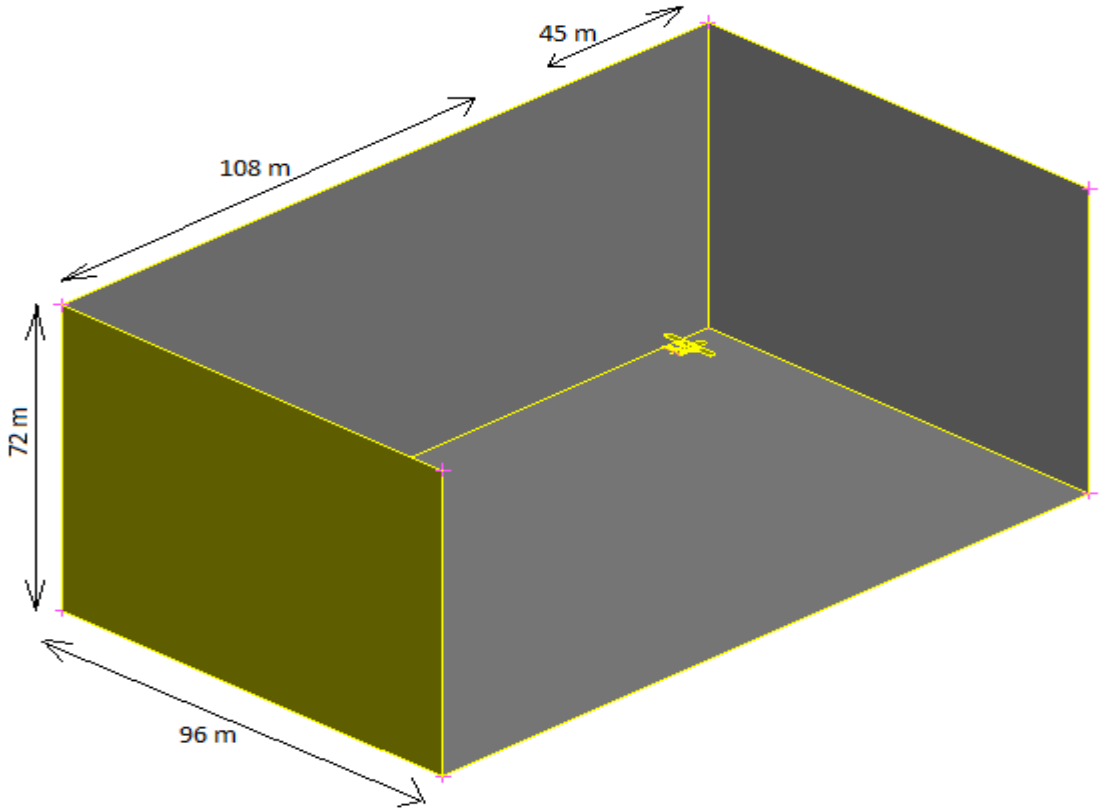


Figura 2.7: Box di calcolo

3 Mesh di superficie

3.1 Generazione della mesh di superficie

La generazione della griglia è stata inizialmente effettuata solo su metà modello, per poi essere simmetrizzata attraverso il comando UTILITIES -> TRANSFORM -> COPY una volta conclusa l'operazione. Con tale comando viene copiata anche la mesh effettuata e ciò ha effettivamente dimezzato i tempi di generazione della mesh di superficie. Si riporta in Figura 3.1 lo screen del comando COPY:

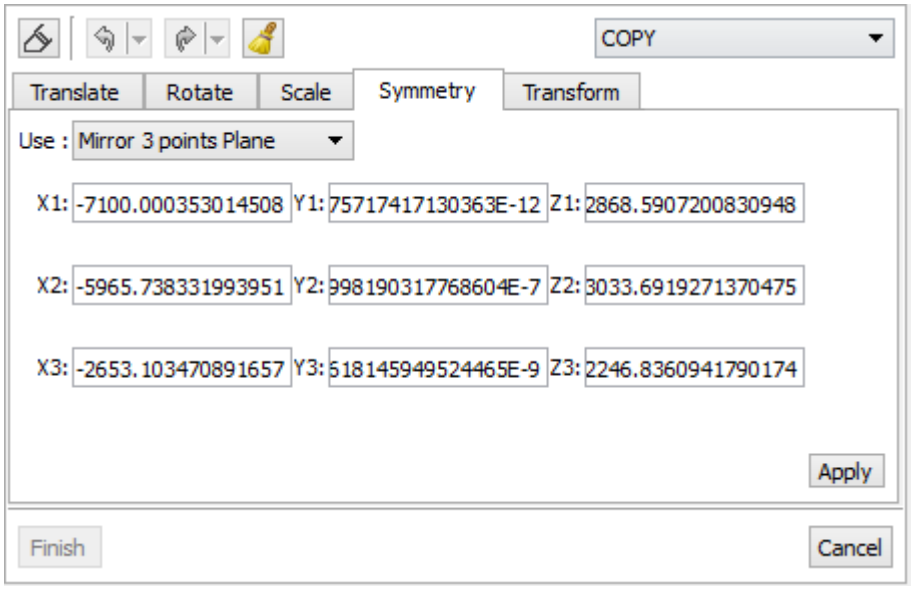


Figura 3.1: Comando COPY

Come si vede bene dalla figura, basta definire il piano di simmetria tramite le coordinate di 3 punti distinti.

Il primo passo effettuato è stato quello di isolare il motore e utilizzare il comando MESH -> PERIMETERS -> LENGTH, che serve per introdurre dei nodi sulle linee ivi presenti a distanza fissata fra loro. Nel caso del motore, si è scelto di imporre una lunghezza di 30 mm. Si è adottato lo stesso procedimento anche per il carrello, le code, la fusoliera posteriore e le pinne imponendo la stessa lunghezza di base adoperata nel motore. Per le suddette parti, si è scelto

l'algoritmo FREE di generazione della griglia. Data la complessità del modello geometrico e l'assenza di una direzione preferenziale di calcolo, si è preferito utilizzare elementi triangolari che risultano essere i più versatili per una corretta definizione delle superfici della griglia di calcolo. Nella Figura 3.2 viene riportato il motore suddiviso in nodi e nella seguente Figura 3.3 la griglia così generata:

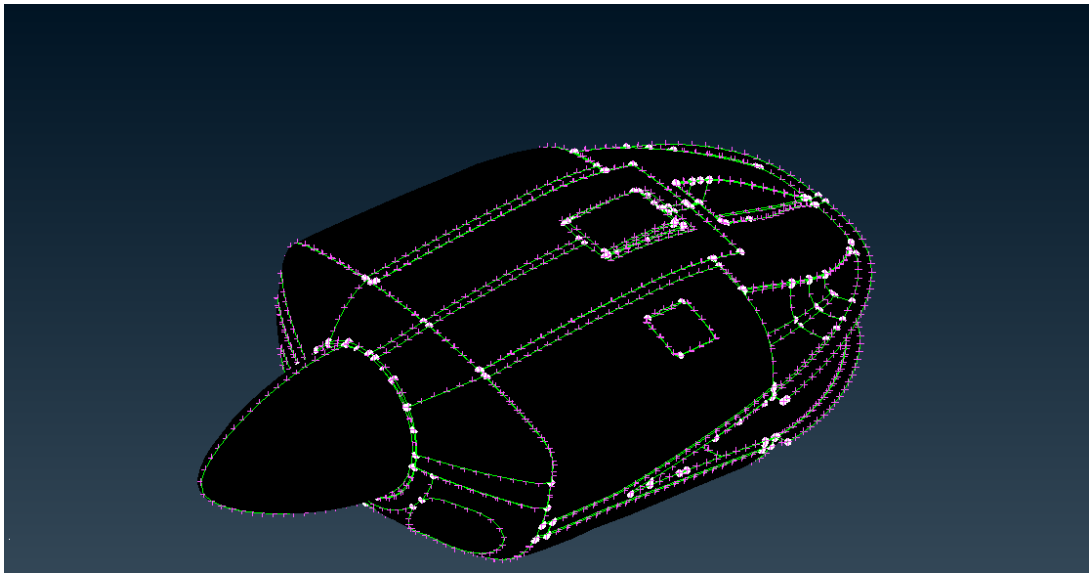


Figura 3.2: Particolare del motore, nodi

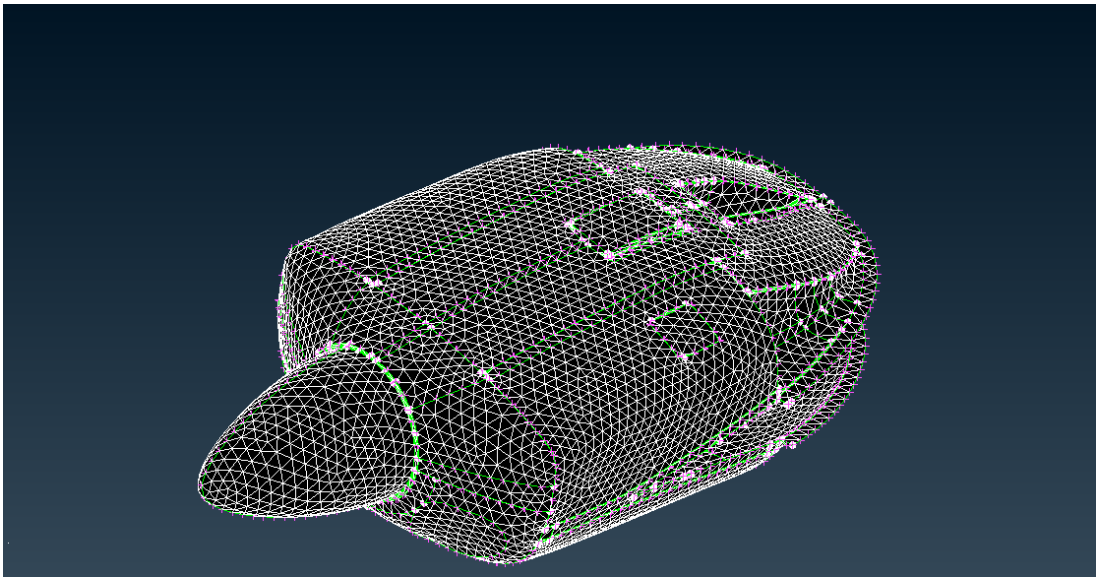
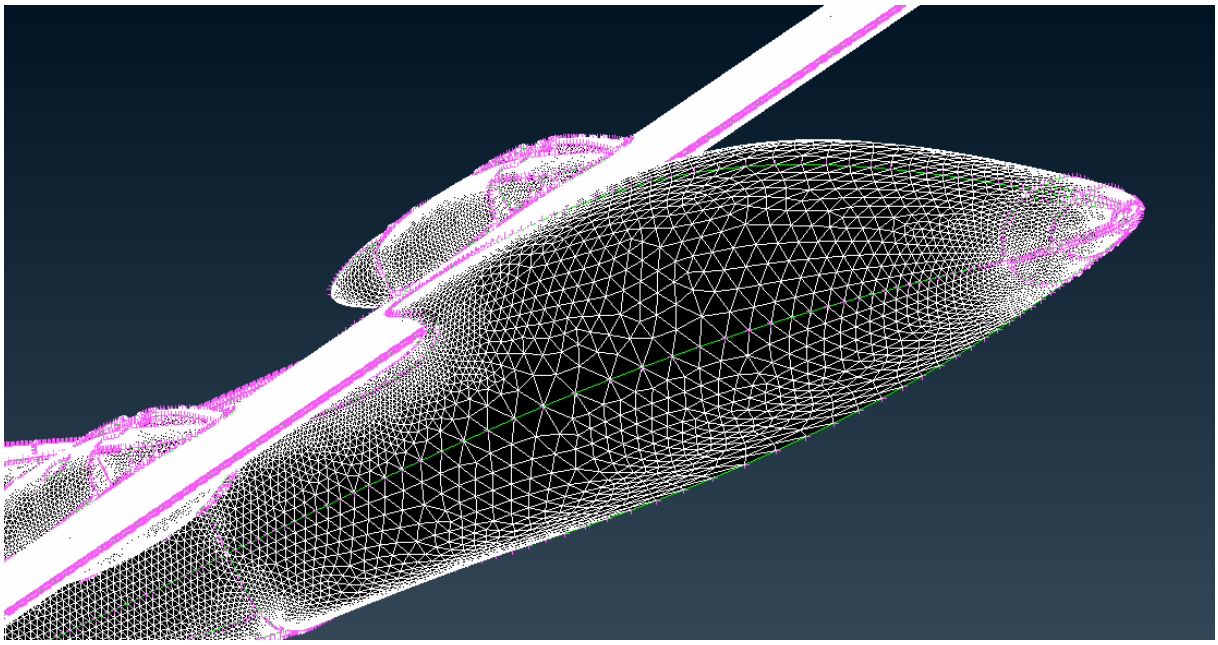


Figura 3.3: Particolare del motore, mesh

L’algoritmo FREE genera una griglia ad elementi triangolari uniforme. Ovvero la dimensione base dell’elemento della mesh non varia lungo la superficie e non è possibile definire un *growth rate*.

Per le restanti parti del velivolo si è preferito usare il comando MESH -> PERIMETERS -> SPACING -> Auto CFD, che suddivide le linee in nodi a spaziatura variabile e dipendenti dalla curvatura locale. Ciò è stato utile sia per raffinare nelle zone ad alta curvatura come ad esempio i bordi d’attacco delle ali, sia per alleggerire la mesh nelle zone a bassa curvatura, come la fusoliera anteriore. Si veda, a fronte di quest’ultima affermazione, la differenza fra generazione della griglia con algoritmo FREE rispetto all’algoritmo CFD usato in accoppiamento con lo *spacing* Auto CFD:



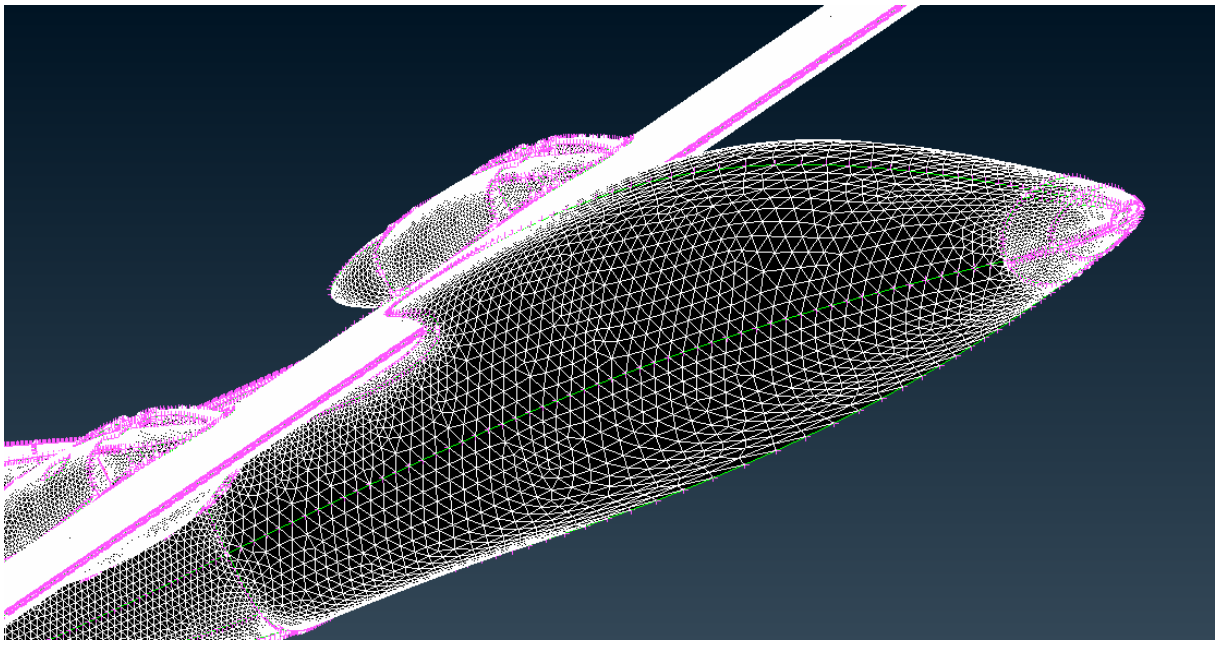


Figura 3.4: Differenze algoritmi di generazione mesh

Tra le impostazioni dell'algoritmo CFD si è imposto un *growth rate* di 1.1 per evitare un'eccessiva disparità di dimensioni degli elementi triangolari che potrebbe inficiare l'accuratezza della soluzione.

Per il dominio di calcolo si è imposto un valore base di LENGTH di 2500 mm.

3.2 Controlli sulla mesh

La fase finale della procedura di generazione della griglia consiste in un controllo selettivo sugli elementi triangolari appena generati. Il fattore discriminante per la qualità della mesh è la *skewness* che rappresenta un indice dell'irregolarità, in senso geometrico, della cella. La *skewness* è così definita:

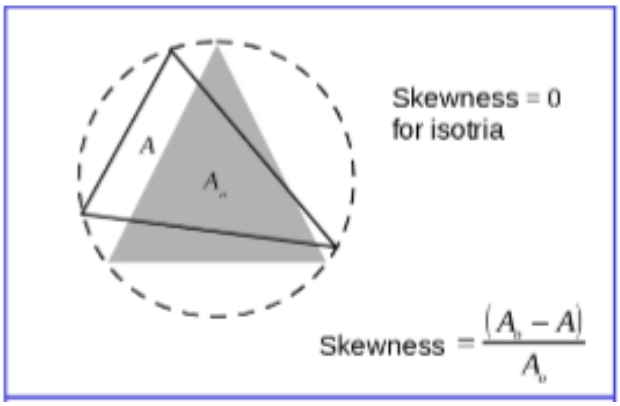


Figura 3.5: Definizione Skewness

Vale 0 per triangoli equilateri e 1 per triangolo degenere. Più ci si avvicina all'unità e più i risultati della simulazione sono soggetti ad errori numerici. Il limite conservativo fissato dall'azienda OMA SUD è 0,75. Qualunque cella con valore di *skewness* maggiore a 0,75 è stata manualmente modificata. I comandi principali utilizzati per lo scopo sono MESH -> MACROS -> CUT e MESH -> MACROS -> JOIN.

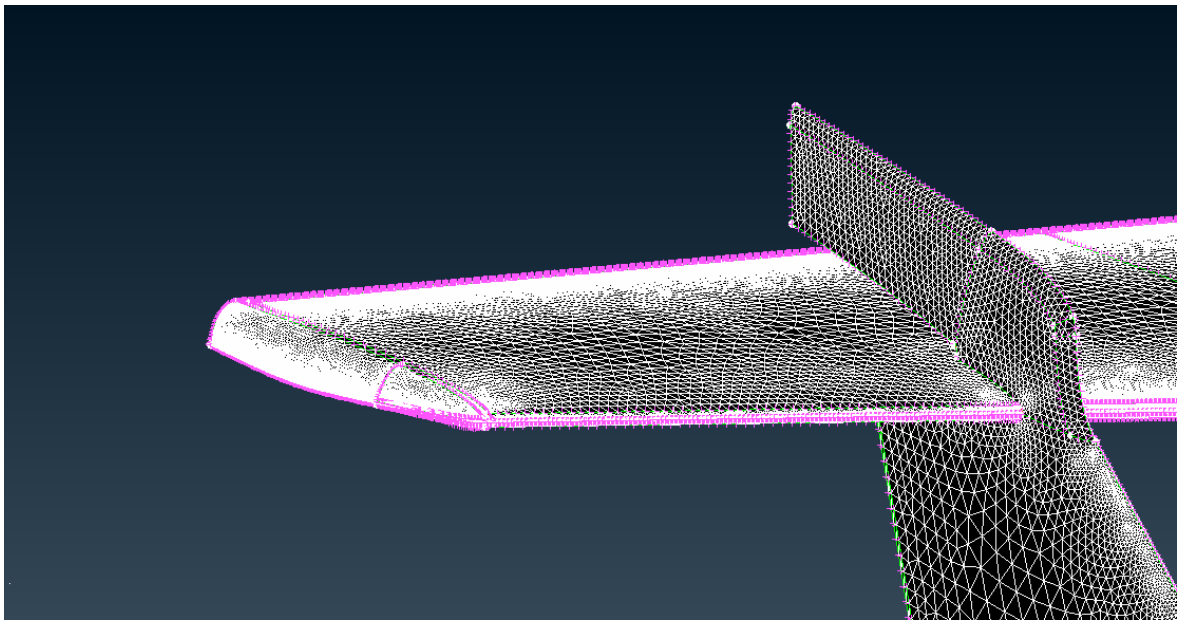
Gli ultimi check prima dell'effettiva conclusione della fase di meshatura superficiale sono:

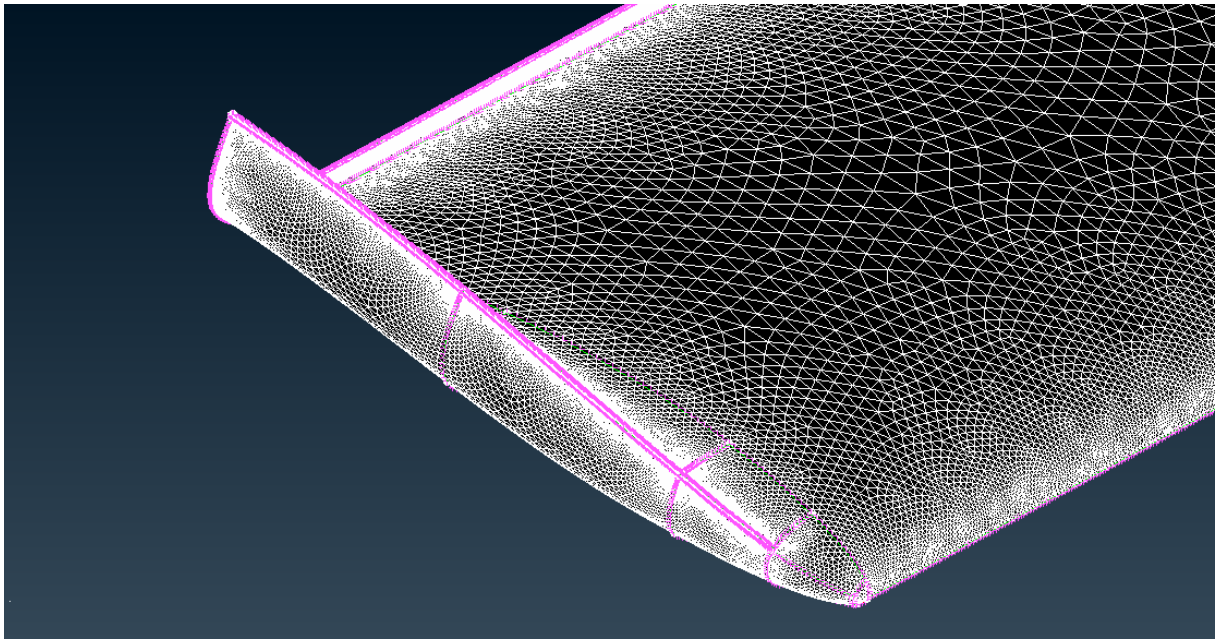
- Unmeshed macros: controlla l'eventuale presenza di elementi nel modello nei quali non è stata generata una griglia
- Duplicate elements: controlla l'eventuale presenza di più celle sovrapposte fra loro. Notare che tale controllo viene effettuato da ANSA sulla base dei valori di tolleranza impostati di default. E' opportuno quindi evitare che la lunghezza minima degli elementi di griglia generati siano di dimensione inferiore ai valori di tolleranza. Se è proprio necessario mantenerli, occorre ridurre le impostazioni di tolleranza.

Vengono riportati in Tabella 2 i dettagli della griglia, relativi al modello completo:

Elementi triangolari	938.602
Elementi quadrati	0
Max Skewness	0,7499

Tabella 2: Dettagli griglia di calcolo





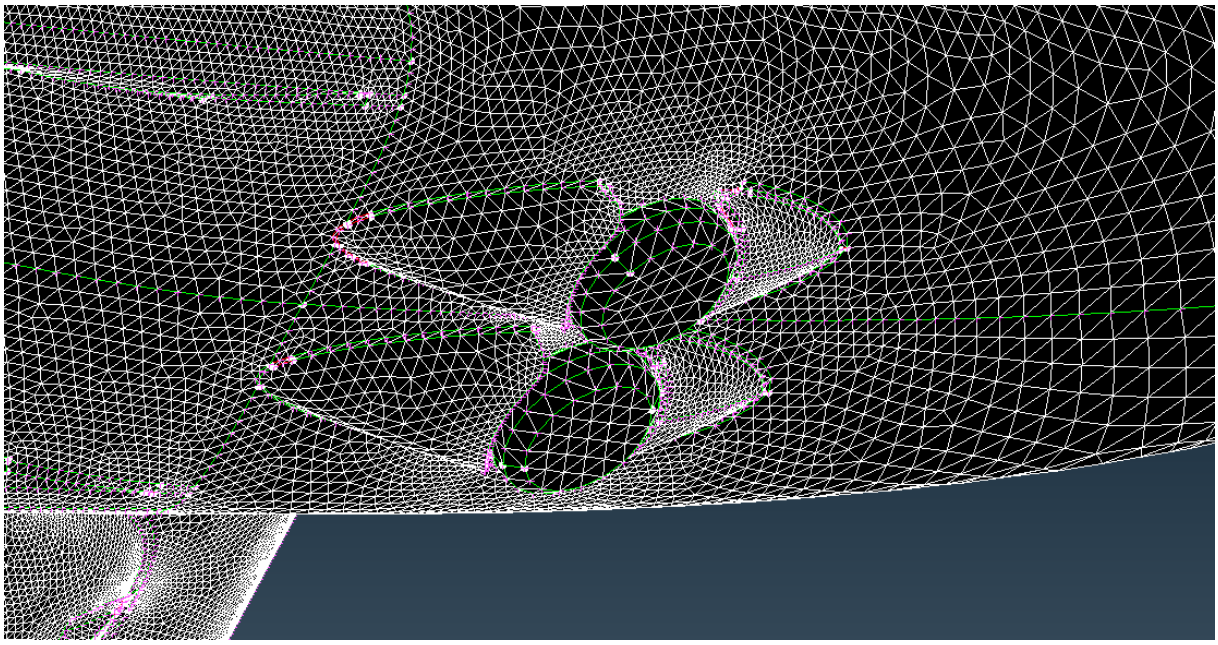


Figura 3.6: Particolari mesh finale

3.3 PID

ANSA permette di suddividere il modello in varie parti tramite l'utilizzo del comando FACES -> SET PID. Ad ognuna di queste parti da noi scelte, è possibile assegnare un nome arbitrario. Ciò torna molto utile nella fase di valutazione dei risultati, come vedremo nei capitoli successivi. Infatti è possibile isolare, ad esempio, la resistenza generata da un singolo PID per valutarne poi l'effettivo contributo alla resistenza totale. In Tabella 3 sono stati riportati i PID con numero di elementi triangolari associati:

Id	Nome	Tipo	Num. Elem.
1	Semiala_SX	Wall	199528
2	Semiala_DX	Wall	201867

3	Winglet_SX	Wall	42958
4	Winglet_DX	Wall	42227
5	Fusoliera_ant_DX	Wall	8144
6	Fusoliera_ant_SX	Wall	8395
7	Muso_DX	Wall	1877
8	Muso_SX	Wall	1897
9	Fusoliera_post_DX	Wall	3326
10	Fusoliera_post_SX	Wall	3278
11	Coda_DX	Wall	6678
12	Pinna_bassa_DX	Wall	1054
13	Impennaggio_vert_DX	Wall	28670
14	Coda_SX	Wall	7192
15	Pinna_bassa_SX	Wall	1178
16	Impennaggio_vert_SX	Wall	28334
17	Impennaggio_oriz_SX	Wall	107962
18	Impennaggio_oriz_DX	Wall	108682
19	Winglet_coda_DX	Wall	29296
20	Winglet_coda_SX	Wall	27740
21	Motore_SX	Wall	19351
22	Motore_DX	Wall	19076
23	Carrello_SX	Wall	1916
24	Carrello_DX	Wall	2053
25	Pinna_alta_DX	Wall	4123
26	Pinna_alta_SX	Wall	4331
27	Symmetry	Wall	20006
28	Inlet	Wall	2526
29	Outlet	Wall	2524
30	Inlet_motore_DX	Wall	530
31	Outlet_motore_DX	Wall	424
32	Outlet_motore_SX	Wall	430
33	Inlet_motore_SX	Wall	530
34	Outlet_up_motore_DX	Wall	207
35	Outlet_up_motore_SX	Wall	292

Tabella 3: Suddivisione PID

La suddivisione in componenti a destra e a sinistra del piano di simmetria del velivolo (DX e SX) è giustificata dal fatto che la simmetria della soluzione non è garantita. Le eliche del velivolo ruotano entrambe nello stesso senso e ciò comporta già di per sé un'asimmetria del flusso. Poiché la configurazione del velivolo concessa dalla OMASUD per la presente tesi risulta non essere in condizione trimmata, si è preferito assumere che le eliche girino in senso antiorario, altrimenti si veniva a creare un momento di rollio che non viene in alcun modo bilanciato. Si è comunque scelto di lasciare inalterata la suddivisione destra/sinistra per testare che effettivamente l'aerodinamica del problema goda di simmetria come è logico aspettarsi.

Nella Figura 3.7 è stata riportata l'immagine del velivolo con l'appena descritta suddivisione in PID:

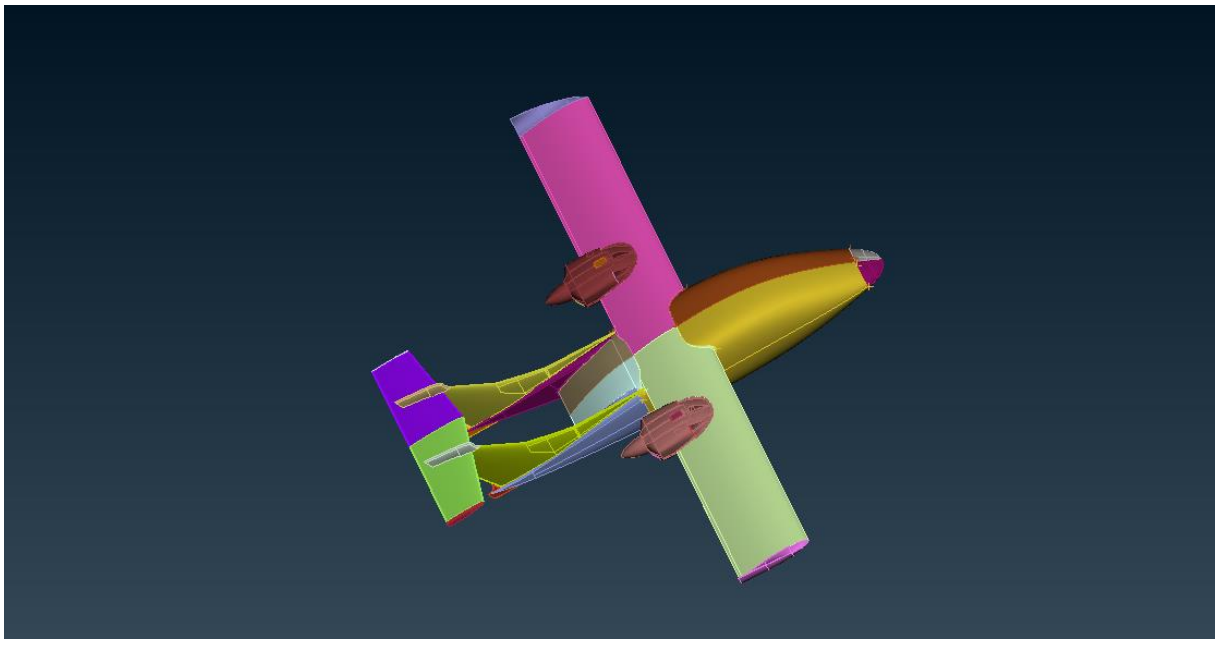


Figura 3.7: Velivolo e suddivisione in PID

4 Mesh di volume

4.1 Importazione del modello

Si è scelto di generare la mesh di volume tramite il software STAR CCM+®. Al fine di una corretta importazione della mesh è necessaria una conversione del formato file della mesh di superficie appena ottenuta in uno compatibile con STAR-CCM+. Tale operazione si svolge in ambiente ANSA con il comando File -> Output -> Star:

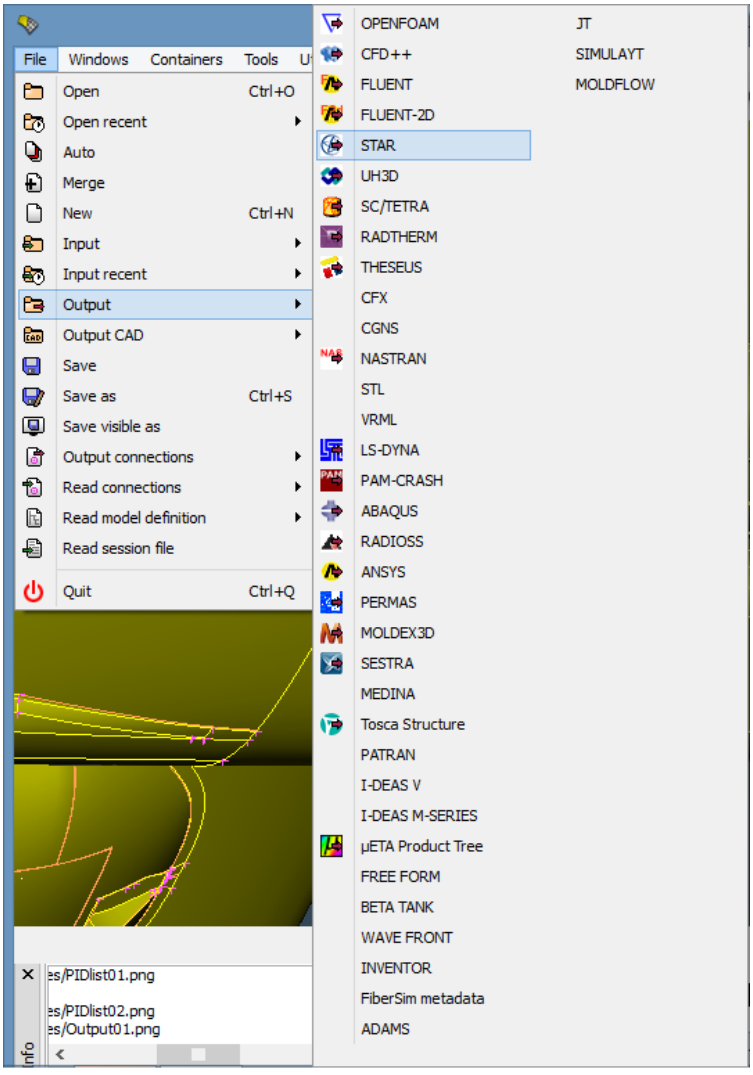


Figura 4.1: Output Star

Viene così creato il file .inp della mesh di superficie. Una volta ottenuta la conversione, si entra in ambiente STAR e, tramite il comando File -> Import -> Import Surface Mesh, si conclude l'importazione del modello. In Figura 4.2 viene mostrato lo screen con le impostazioni usate in STAR per l'apertura del file:

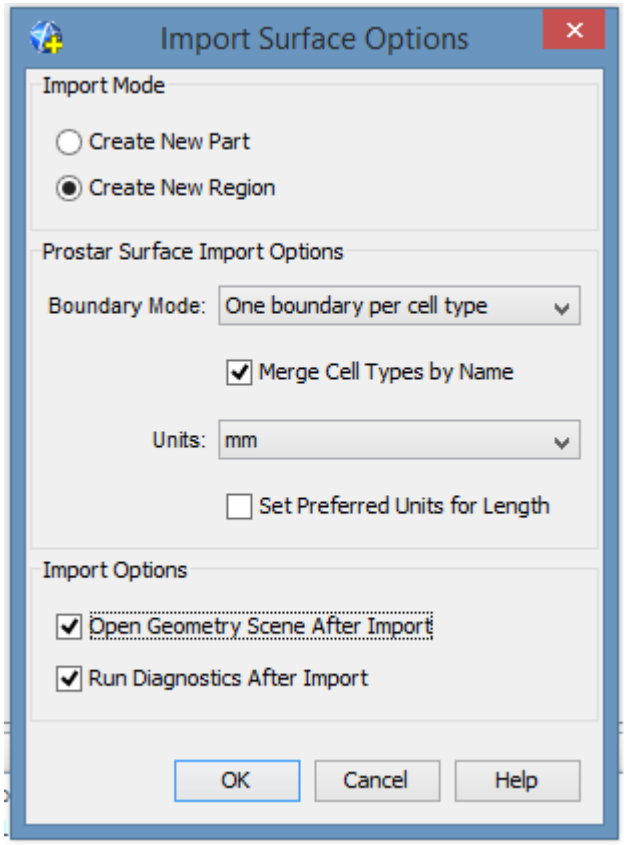


Figura 4.2: Finestra "Import Surface Option"

Tutta la geometria del velivolo con annesso il dominio di calcolo è racchiusa in un'unica *Region* e i PID che si sono imposti in ambiente ANSA vengono automaticamente settati come *Boundaries*.

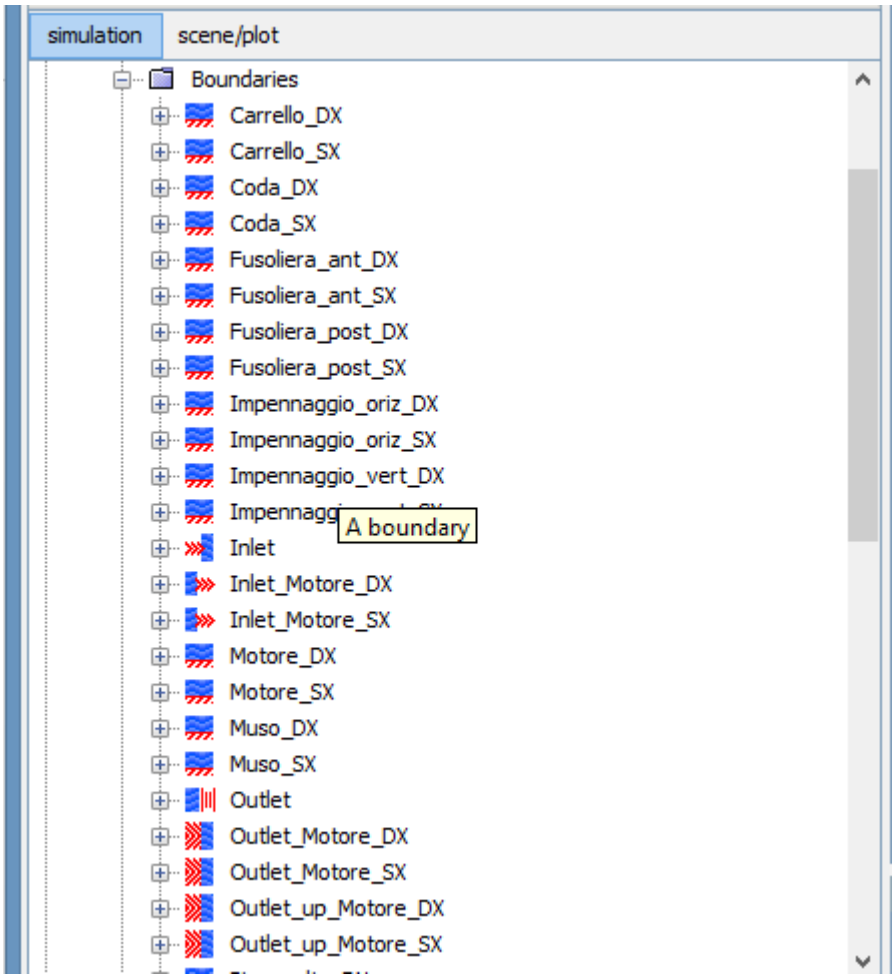


Figura 4.3: Boundaries

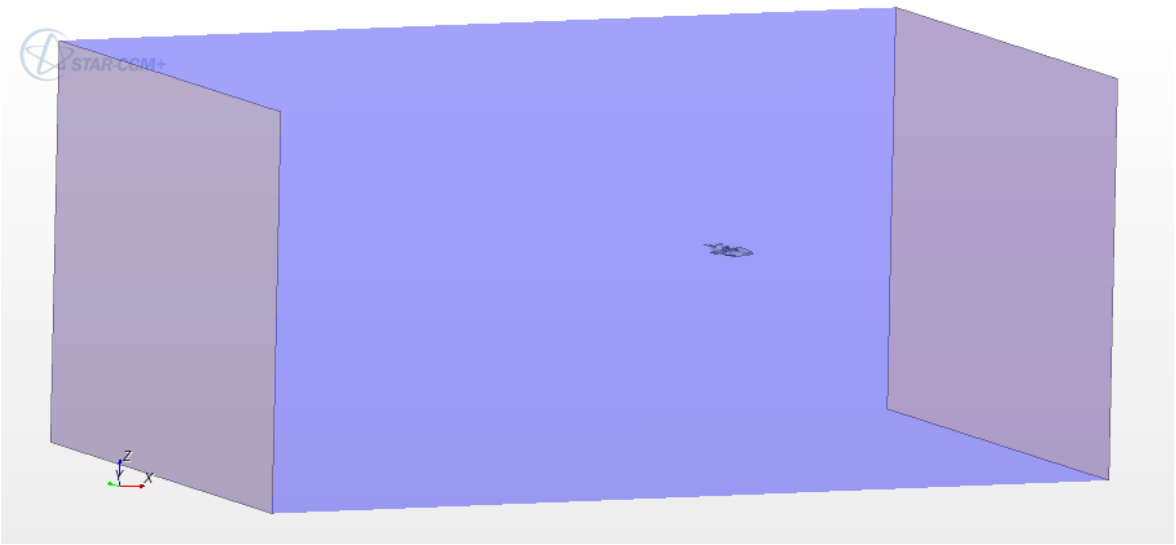


Figura 4.4: Importazione CAD

4.2 Generazione della mesh di volume

I modelli utilizzati per la griglia di volume sono:

- *Polyhedral Mesh*: suddivide il dominio in celle poliedriche ottenute per combinazione a partire da celle tetraedriche. La generazione di tale celle richiede un costo computazionale maggiore rispetto agli altri modelli, ma garantiscono una soluzione più accurata soprattutto per domini complessi come il caso in esame.
- *Prism Layer Mesher*: genera strati di celle prismatiche attorno alle superfici del modello diretti ortogonalmente ad esse. Serve per ottenere una descrizione più accurata dello strato limite turbolento e delle forze viscose.

L'applicazione del modello *Prism Layer Mesher* è stata disabilitata nelle superfici che compongono il box esterno al velivolo e nelle zone di ingresso/uscita di portata d'aria nei motori.

Dopo varie prove di generazione della mesh, sono stati modificati alcuni parametri caratteristici impostati di default all'interno del ramo *Continua -> Mesh*. Essi sono:

Number of Prism Layers: stabilisce il numero di strati di celle prismatiche

Prism Layer Stretching: imposta il rateo di crescita in spessore di ogni strato di celle prismatiche rispetto allo strato precedente

Prism Layer Thickness: stabilisce lo spessore complessivo della zona a celle prismatiche

Tet/Poly Density: controlla la densità e il fattore di crescita delle celle globali attraverso 2 parametri. Il primo, *Density*, regola la densità delle celle; un valore superiore a 1 la aumenta e un valore inferiore a 1 la diminuisce. Il secondo, *Growth Factor*, agisce sulla velocità di crescita delle dimensione delle celle nel passaggio dalle zone in cui la griglia è raffinata al nucleo della mesh di volume. Se il valore impostato è maggiore di 1 la crescita è più rapida rispetto al valore di default, viceversa in caso sia minore di 1, il che implica la generazione di un maggior numero di celle.

Si riporta in Figura 4.5 lo screen con tutte le impostazioni modificate per il caso:

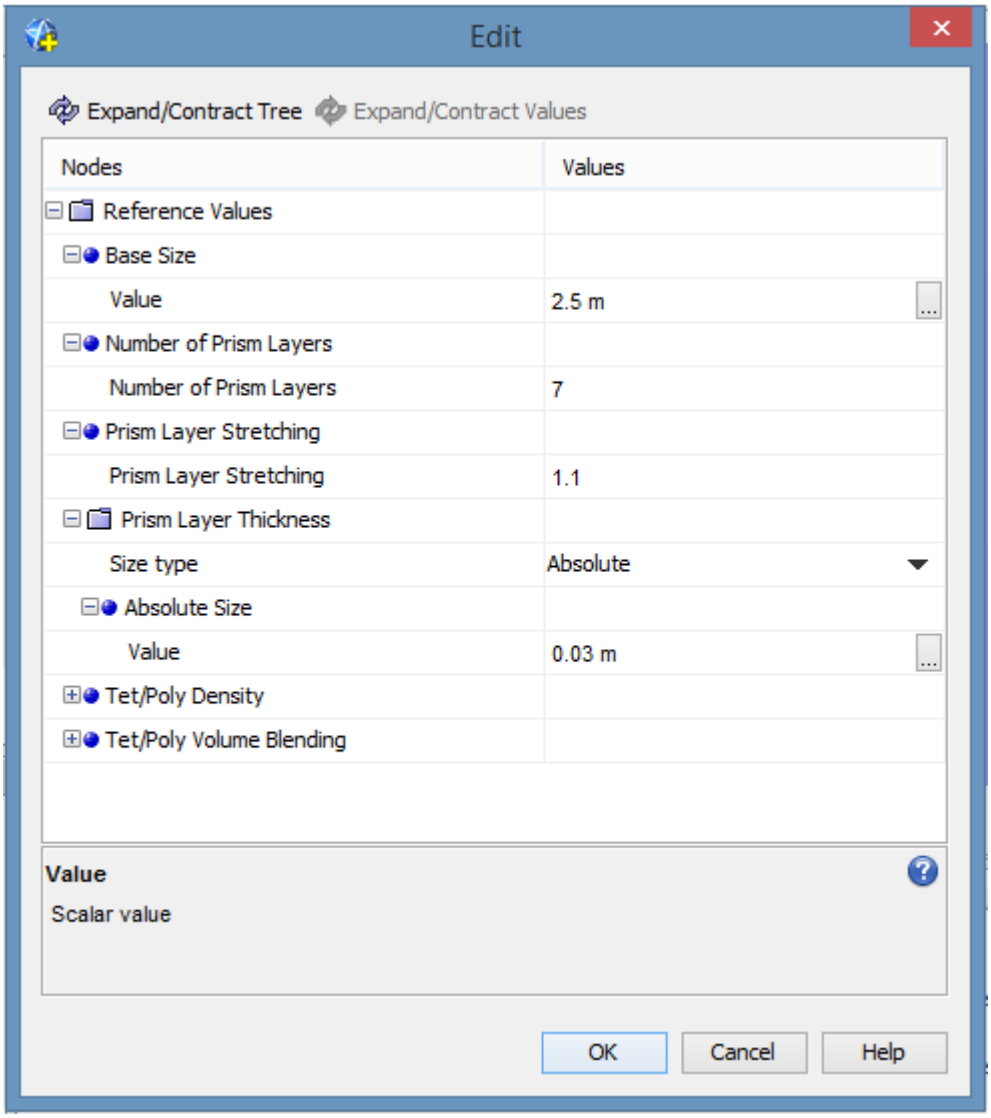


Figura 4.5: Settaggi Mesh di volume

Il valore di 3 cm per lo spessore complessivo del *Prism Layer*, è stato valutato con l'analogia della lastra piana, prendendo come riferimento la corda del profilo alare e utilizzando la seguente formula:

$$\left(\frac{\delta}{L}\right)_{turb} = \frac{0.377}{(Re_L)^{\frac{1}{5}}}$$

Per valutare lo spessore dello strato limite turbolento δ_{turb} si è per semplicità applicata una legge approssimata, dedotta su basi sperimentali detta "legge delle potenze". Il risultato che si

ottiene, facendo l'approssimazione non veritiera, ma conservativa, che la transizione laminare/turbolento avvenga al bordo d'attacco, è:

$$\delta_{turb} \cong 2.4\ cm$$

Approssimato nel modello a 3 cm per essere sicuri di analizzare al meglio lo strato limite.

Si è scelto di applicare il *Prism Layer Mesher* solo nelle pareti "solide" del velivolo, escludendo quindi le zone di ingresso/uscita portata d'aria nelle gondole motori e tutte le superfici del dominio di calcolo. Il software Star CCM+ genererà quindi una zona attorno ai *Boundaries* prescelti di spessore totale 3 cm suddivisi in 7 strati ognuno 1,1 volte più grande del precedente. Con i parametri così impostati si assicura un'attendibile simulazione dello strato limite attorno alle superfici.

In aggiunta a queste modifiche si è scelto di creare due zone cilindriche attorno alle gondole motore dove la mesh è stata opportunamente raffinata in accordo con la guida del software per l'applicazione del modello *Virtual Disk Model* che vedremo nel dettaglio successivamente. Per generare le zone cilindriche di raffinamento si utilizza il comando *Geometry -> Parts -> New Shape Parts*.

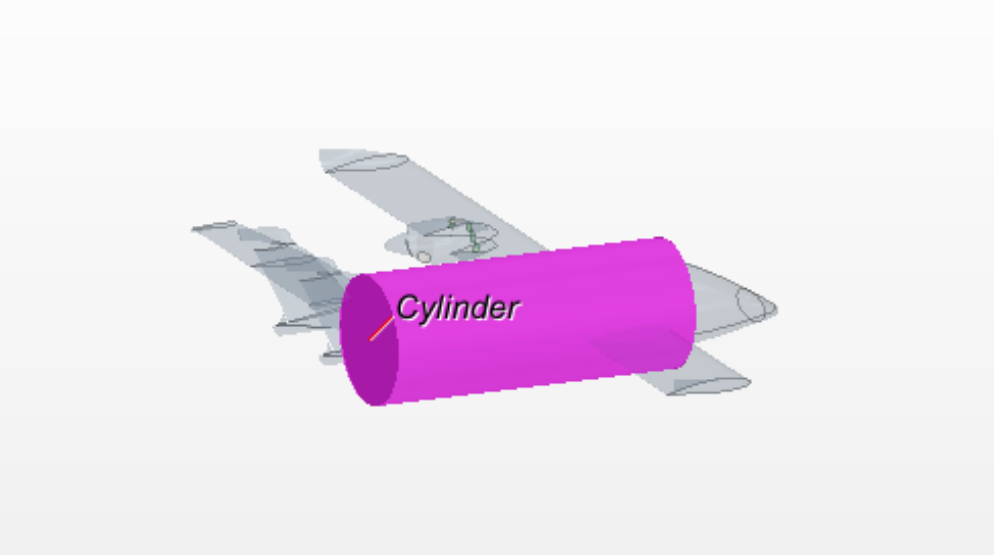
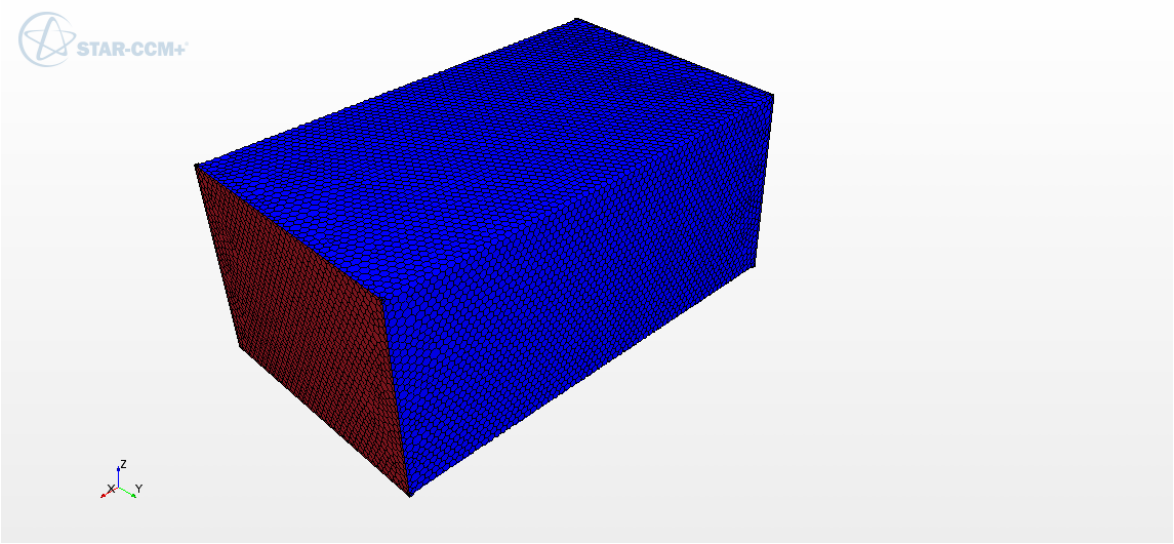
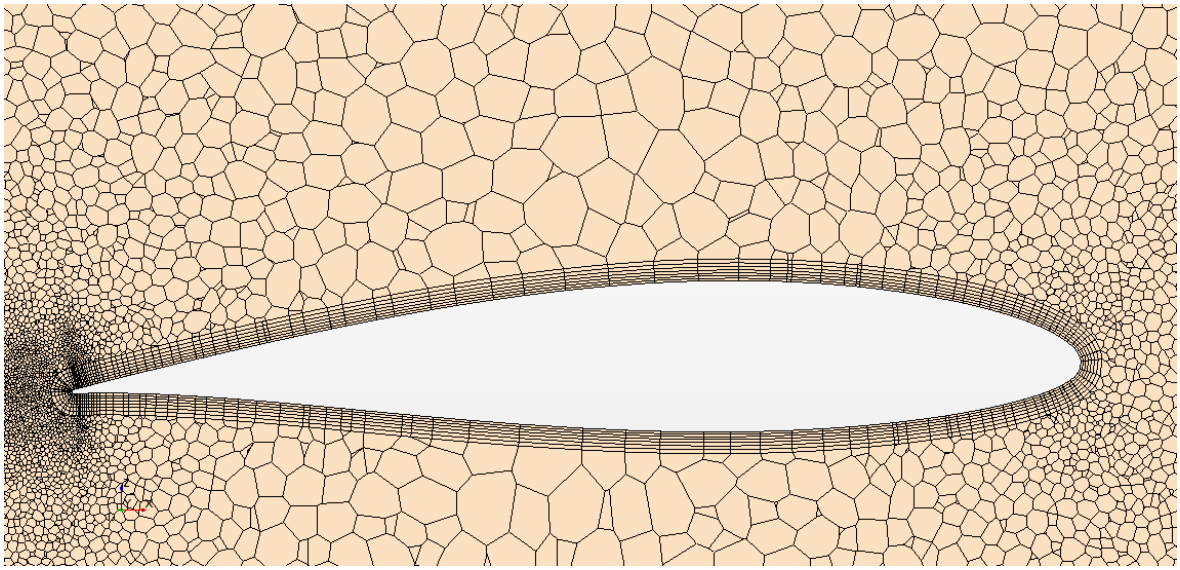


Figura 4.6: Zona di raffinamento in Star CCM+

Successivamente, utilizzando il comando *Continua -> Mesh -> Vometric Controls -> New -> Mesh Values*, si è impostata la dimensione massima della cella di volume a 10 cm. Tale scelta è stata validata a seguito di varie prove in modo tale da ottenere un numero di celle di volume complessive non troppo elevato, che comporterebbe un alto costo computazionale, e assicurando, contemporaneamente, un livello di raffinamento accettabile.

La mesh di volume così generata conta circa 37 milioni di celle.





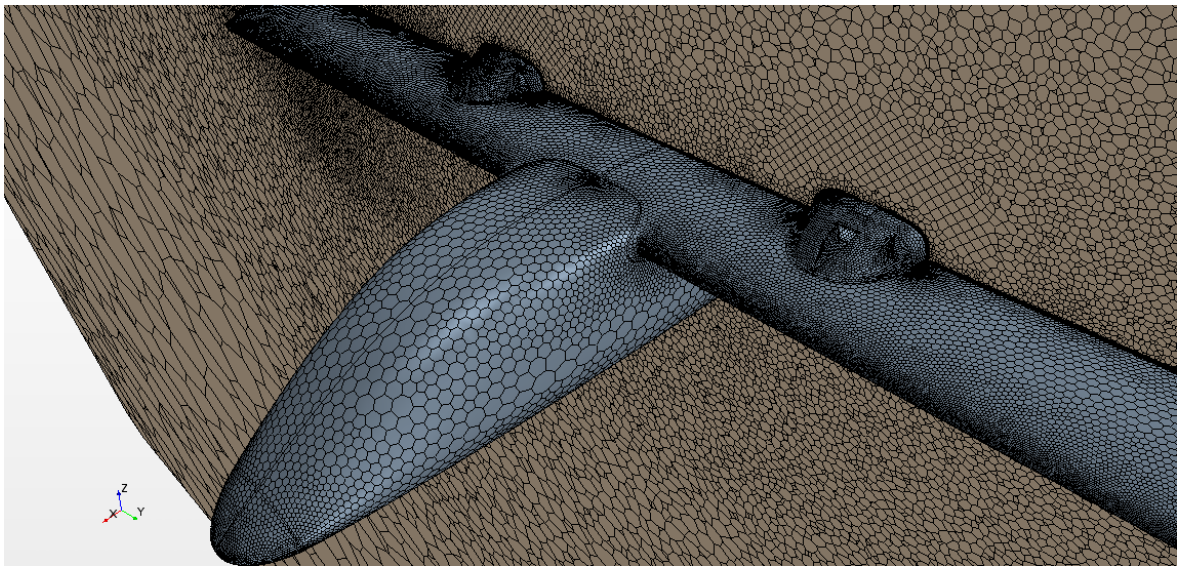


Figura 4.7: Mesh di volume

4.3 Qualità della mesh

Così come si è fatto in ambiente ANSA, è opportuno fare un controllo qualità sulla mesh appena generata.

Tale operazione può essere svolta direttamente in ambiente STAR-CCM+, tramite il comando

- *Mesh --> Diagnostic --> Report Type: Full*

che valuta i seguenti parametri:

- *Angolo di Skewness* (θ): è l'angolo tra la retta congiungente i centroidi di due celle adiacenti e la direzione normale alla faccia comune; un angolo di 0° indica una mesh perfettamente ortogonale, un angolo maggiore di 90° indica celle concave e ciò può dar luogo a problemi di convergenza. E' stato imposto dunque un limite massimo di 85° superato il quale la mesh è da rifare.
- *Volume Change*: descrive il rapporto tra il volume di una cella è quella della più grande cella ad essa adiacente. Un valore di 1.0 indica che il volume della cella è maggiore o uguale a quello di quelle confinanti, mentre discontinuità troppo alte sono indice di problemi di convergenza. In ambiente Star CCM+ è possibile individuare tali celle grazie alla funzione di campo *Bad Cell Indicator*. In ogni caso, si impone il valore minimo, fornito nella guida del software, di 10^{-5}.

- *Face Validity*: misura l'orientamento delle normali alle facce rispetto al centroide della cella. Un valore di 1.0 indica che tutte le normali alle facce sono dirette in direzione opposta al centroide, mentre valori al di sotto dell'unità indicano la presenza di concavità nella cella.

Nella Tabella 4 si riportano i valori ottenuti per il caso in esame. Come si vede, i valori massimi (o minimi) riscontrati rientrano nei valori di tolleranza imposti:

Maximum Sknewness	83.2°	<85°
Minimum Volume Change	$9.723 \cdot 10^{-4}$	$>10^{-5}$
Minimum Face Validity	0.7	>0.5

Tabella 4: Controllo qualità mesh di volume

Da notare, per quanto riguarda il Minimum Face Validity, che il valore di 0.7 è stato ottenuto solo nello 0.01% delle facce totali delle celle. Il 98.1%, invece, presenta il valore di Minimum Face Validity pari a 1.0.

5 Settaggi simulazione

5.1 Condizioni iniziali e al contorno

Una volta ottenuta la griglia di calcolo, il passo successivo è quello di ottenere i coefficienti aerodinamici agenti sul velivolo. Occorre dunque settare le opportune condizioni iniziali e al contorno e i modelli fisici da utilizzare.

All'interno dei rami *Physics -> Reference Value* e *Physics -> Initial Conditions* sono stati impostati rispettivamente i valori di pressione atmosferica alla quota di crociera (4000 ft) e velocità (140 kts), lasciando invariati i valori riguardanti la turbolenza. Il settaggio viene riportato in Tabella 5:

Pressione di riferimento *(Reference value)*	87514 Pa
Pressione *(Initial conditions)*	0.0 Pa
Velocità asintotica	72 m/s
Densità	1.088 kg/m^3
Viscosità dinamica	1.77223E-5 Pa-s

Tabella 5: Condizioni iniziali

Si noti che il valore di pressione in *Physics -> Initial Conditions* è espresso in *gauge* ovvero come valore da aggiungere/sottrare al valore di riferimento imposto in *Physics->Reference Value.* La velocità espressa in componenti, tiene conto dell'incidenza del velivolo in crociera pari a 3 deg.

Le condizioni al contorno vanno settate su ogni *Boundary* del modello importato. Si ha così:

Velocity Inlet: si applica alla faccia anteriore del dominio di calcolo (denominata Inlet) e serve per settare la velocità in ingresso nel dominio.

Pressure Outlet: si applica alla faccia posteriore del dominio di calcolo (denominata Outlet) e serve per settare la condizione di pressione in uscita necessaria per l'integrazione

Symmetry: si applica alle facce laterali del dominio di calcolo (che compongono un'unica *Boundary* denominata Symmetry) e serve per azzerare le tensioni tangenziali

Flow-Split Outlet / Mass Flow Inlet: si applicano rispettivamente nelle zone di ingresso e uscita d'aria nelle gondole motore e serve per settare il valore di portata d'aria. Si noti che il dominio sul quale il solutore opera è quello compreso fra box e velivolo, è per questo che nella zona di ingresso del motore va settato un Flow-Split Outlet e viceversa per la zona di uscita.

Wall: si applica in tutte le restanti superfici del velivolo e serve per imporre la condizione di aderenza

5.2 Definizione del modello fisico

In *Continua -> Physics -> Models* si impostano i modelli fisici scelti per il caso in esame. Essi sono:

- Three dimensional
- Steady
- Gas
- Segregated Flow
- Constant density
- Turbulent
- K-Epsilon Turbulence
- Virtual Disk

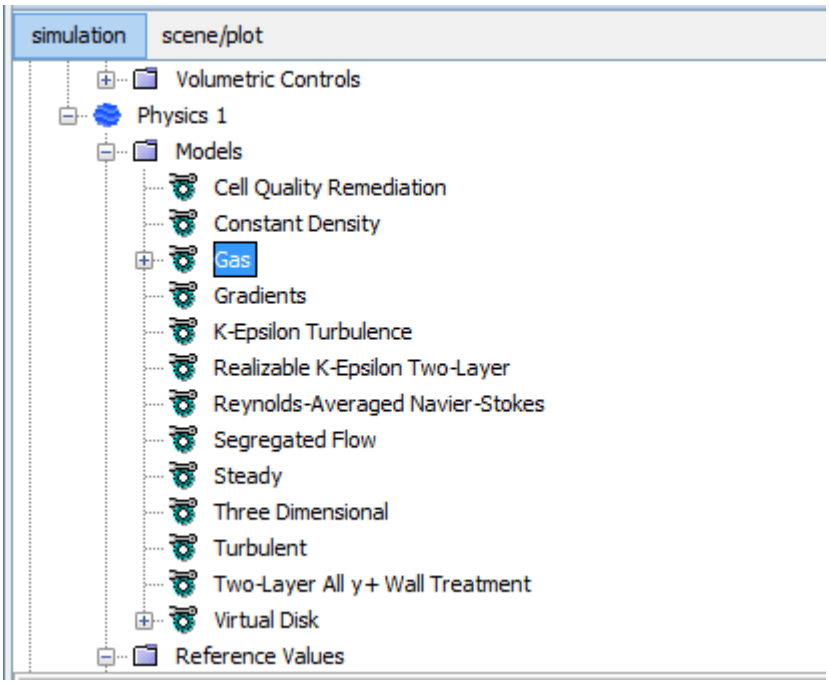


Figura 5.1: Scelta dei modelli fisici

La scelta dei modelli Segregated Flow e Constant Density è giustificata dall'assunzione di flusso incomprimibile, il Mach di crociera del velivolo risulta infatti essere M=0,2. In flusso incomprimibile, l'equazione di stato si riduce a ρ = *costante* e le equazioni di bilancio di massa e bilancio di quantità di moto possono essere disaccoppiate dall'equazione di energia, che, per il caso in esame, non verrà risolta, riducendo così i tempi complessivi di calcolo.

Poiché STAR CCM+ risolve le equazioni di Navier-Stokes mediante RANS (Reynolds Averaged Navier-Stokes), occorre scegliere un opportuno modello di turbolenza. Si stato utilizzato il K-Epsilon Realizable, che lega in modo proporzionale l'energia cinetica turbolenta k e il gradiente di dissipazione ϵ. Tale scelta è giustificata non solo per l'ampia validazione del modello, ma anche perchè offre un buon compromesso tra robustezza, costo computazionale e accuratezza. Non a caso è il modello di turbolenza che solitamente viene utilizzato per applicazioni industriali. In particolare si è deciso di scegliere la versione *Realizable* del modello, che rispetto a quella *Standard*, utilizza una diversa formulazione della viscosità turbolenta e una nuova equazione del rateo di dissipazione ε.

Per simulare l'effetto aerodinamico dell'elica si è usato il *Virtual Disk Model* che si basa sul principio di rappresentare le eliche come un disco attuatore. Per sfruttare al meglio le potenzialità di questo modello è necessario conoscere la curva di prestazione dell'elica e quindi i coefficienti di spinta, di potenza e il rendimento in funzione del coefficiente di avanzamento di pala J. E' opportuno inoltre raffinare la mesh, come già precedentemente detto, in una zona

cilindrica di diametro pari al diametro dell'elica maggiorato del 10% per tenere conto dell'incremento dell'area di scia, e di lunghezza pari a 5 volte il diametro, a valle dell'elica.

Per importare correttamente in ambiente Star CCM i valori riguardanti le prestazioni dell'elica, è stata dapprima realizzata una tabella tramite foglio Excel. Poi, dal ramo *Tools -> Tables*, è stato aggiunto il file. Nella Figura 5.2 viene mostrato lo screen con le impostazioni principali da definire per l'applicazione del modello:

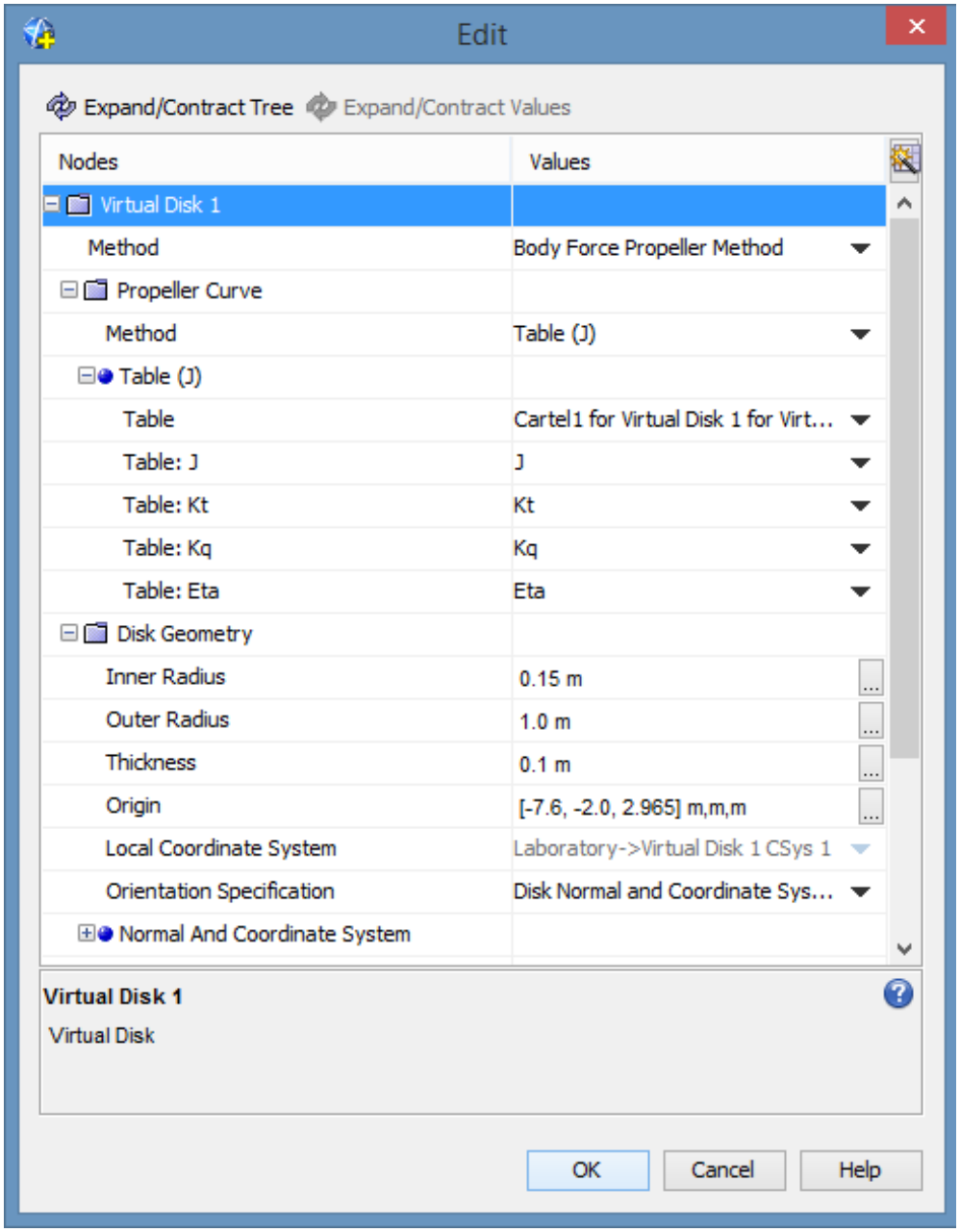


Figura 5.2: Settaggio Virtual Disk Model

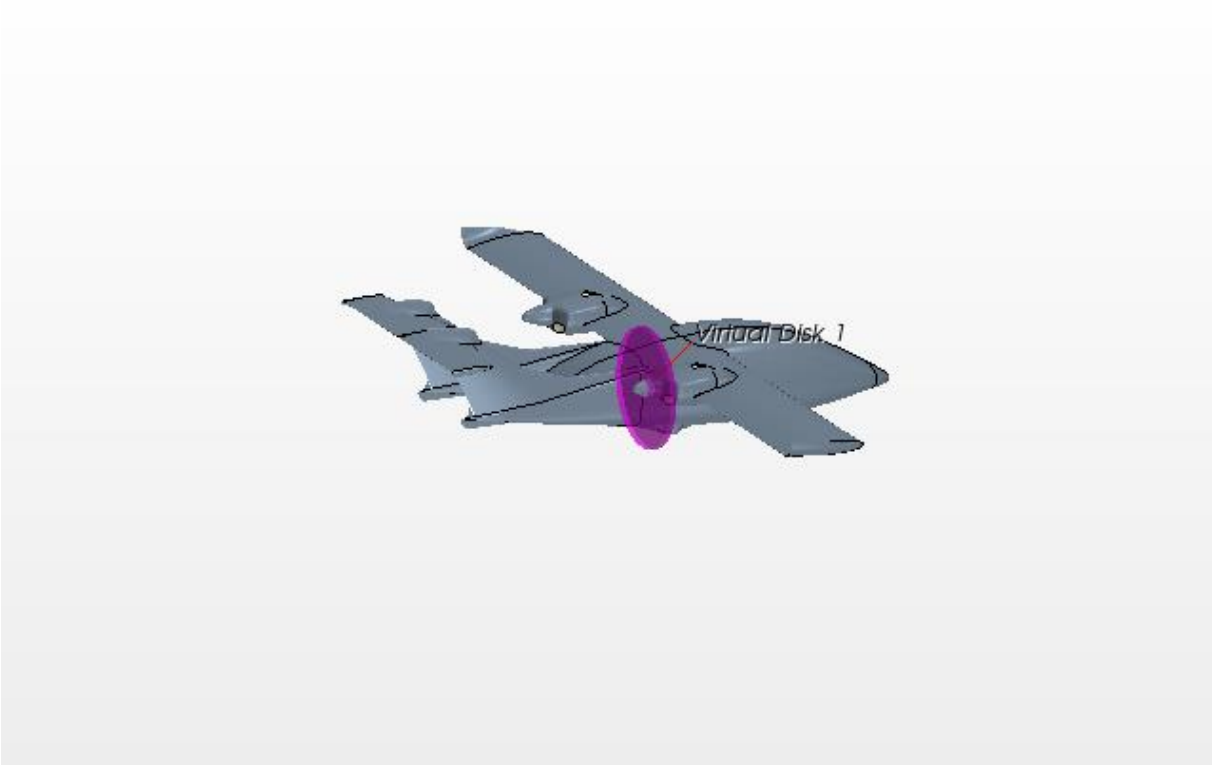


Figura 5.3: Virtual Disk Model

5.3 Settaggi del solutore e report

La fase finale comprende la realizzazione dei *report* delle grandezze fisiche che si è interessati a calcolare. Nel caso in esame si è focalizzata l'attenzione solo sul coefficiente di resistenza. In realtà, i *report* sul coefficiente di portanza e sul coefficiente di momento di beccheggio sono stati realizzati, e da questi è emerso immediatamente che la geometria del modello analizzata non era in condizioni trimmate. Avendo comunque solo questa a disposizione, si è scelto di non considerare i valori di tali *report*.

Infine si è impostato il numero massimo di iterazioni mediante il comando *Stopping Criteria -> Maximum Step*. In modo conservativo, si è imposto un valore limite di 3000 iterazioni, abbastanza elevato da consentire la convergenza per il caso in esame e contemporaneamente abbastanza basso per non inficiare eccessivamente sul costo computazionale. In Figura 5.4 sono riportati i settaggi utilizzati per la creazione del report

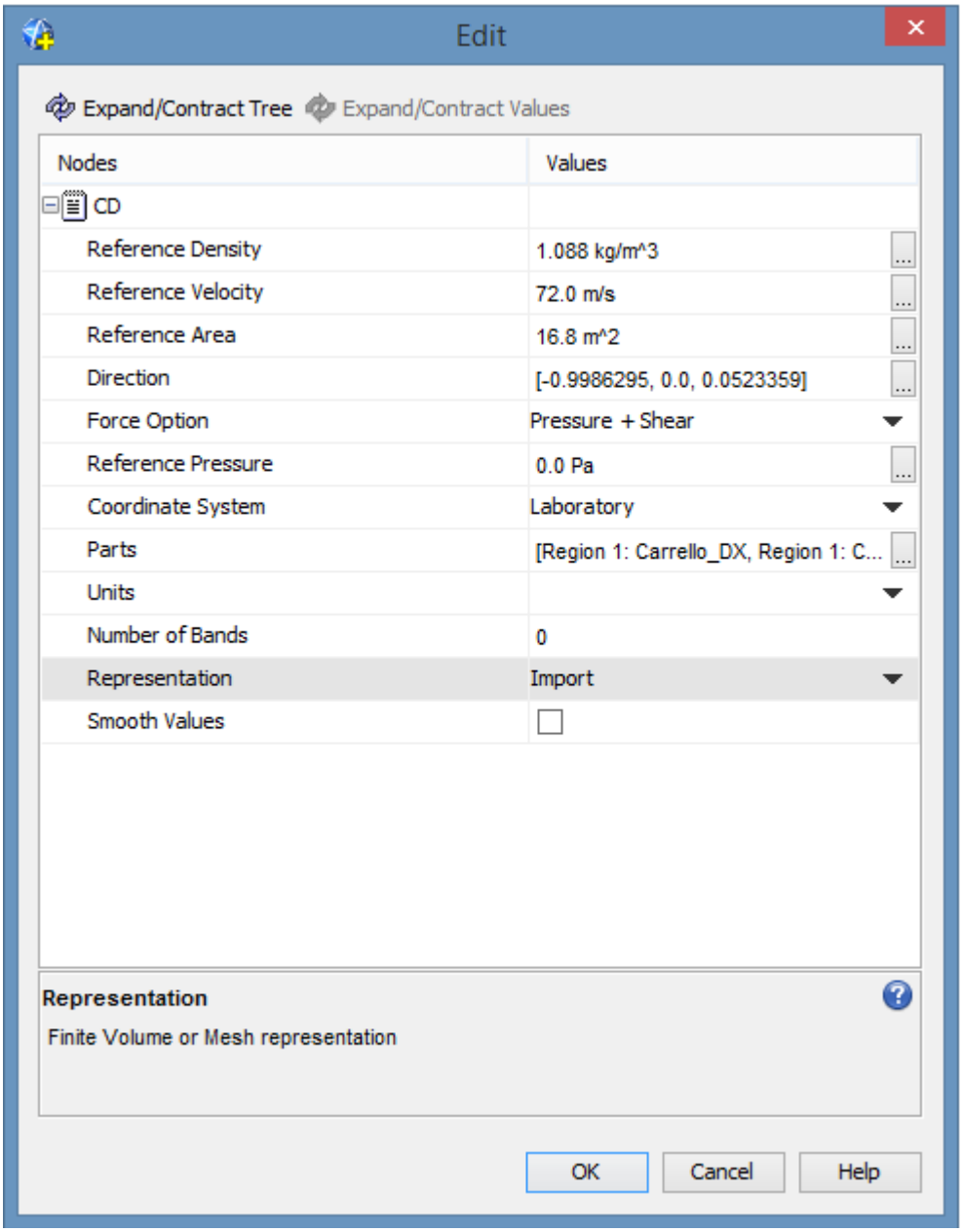


Figura 5.4: Settaggi Report

Tutte le operazioni svolte finora all'interno del software STAR-CCM+® possono essere registrate in una macro, in formato *.java*, tramite il comando *Macro -> Start Recording*, per poter poi essere riprodotte in automatico tramite il comando *Macro -> Play Macro*. In appendice verrà riportato nel dettaglio la struttura delle *Macro* utilizzate. Una piccola nota è a questo punto necessaria per quanto riguarda la *Macro* del settaggio del solutore. Il software messo a disposizione dall'università è Star CCM+ v 8, mentre quello installato nel *cluster* è la versione 10. Quasi tutte le righe di codice della *Macro* sono compatibili, tranne quella che

riguarda il settaggio del *Virtual Disk Model* e in particolare la trasposizione dei coefficienti di coefficienti di spinta, di trazione e il rendimento dalla tabella al solutore. Non avendo a disposizione la guida di Star CCM+ v10 si è dovuto impostare manualmente tale settaggio.

6 Analisi dei risultati

6.1 Definizioni preliminari

Una volta terminato il calcolo, i coefficienti monitorati sono stati importati in un file di testo. I valori di interesse per il caso in esame sono:

$$C_D = \frac{D}{\frac{1}{2}\rho S U_\infty^2} \qquad C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2}$$

I valori di riferimento con cui sono stati calcolati i coefficienti sono espressi nella Tabella 6

Densità (alla quota di crociera)	1,088 kg/m^3
Superficie di riferimento	16,8 m^2
Velocità asintotica	72 m/s
Pressione (alla quota di crociera)	87514 Pa

Tabella 6: Valori di riferimento

6.2 Risultati

Dall'analisi CFD sono stati riportati i valori del coefficiente di resistenza suddivisi per ogni PID, così da trovare più facilmente quale parte del velivolo dava il maggior contributo alla resistenza totale.

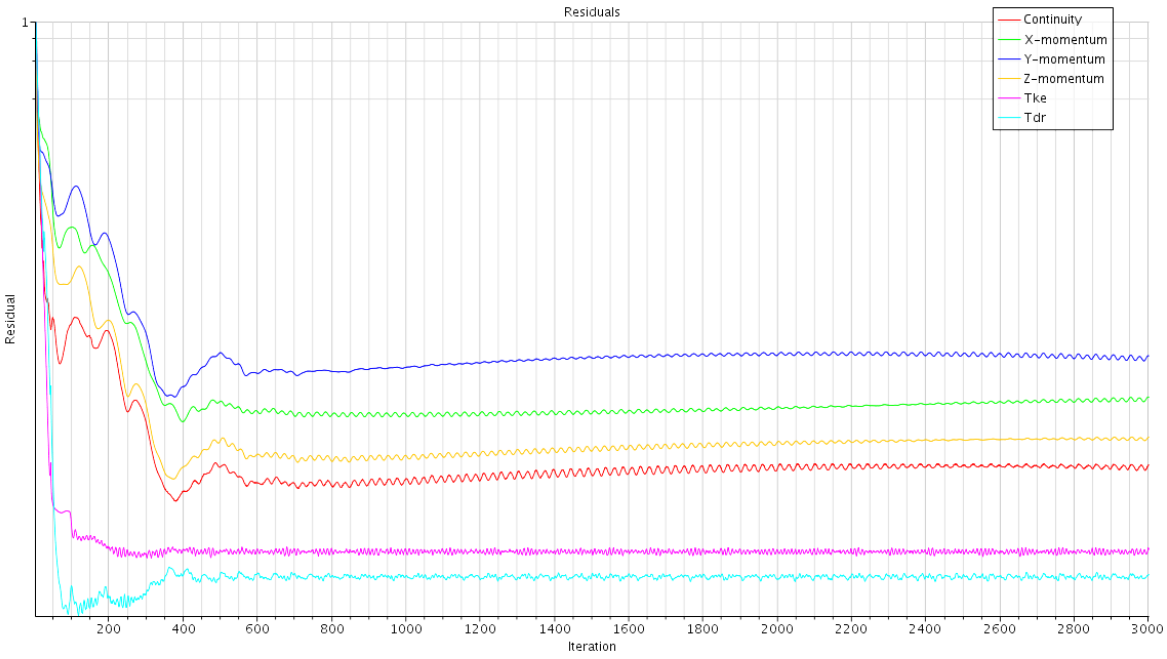


Figura 6.1: Grafico dei residui

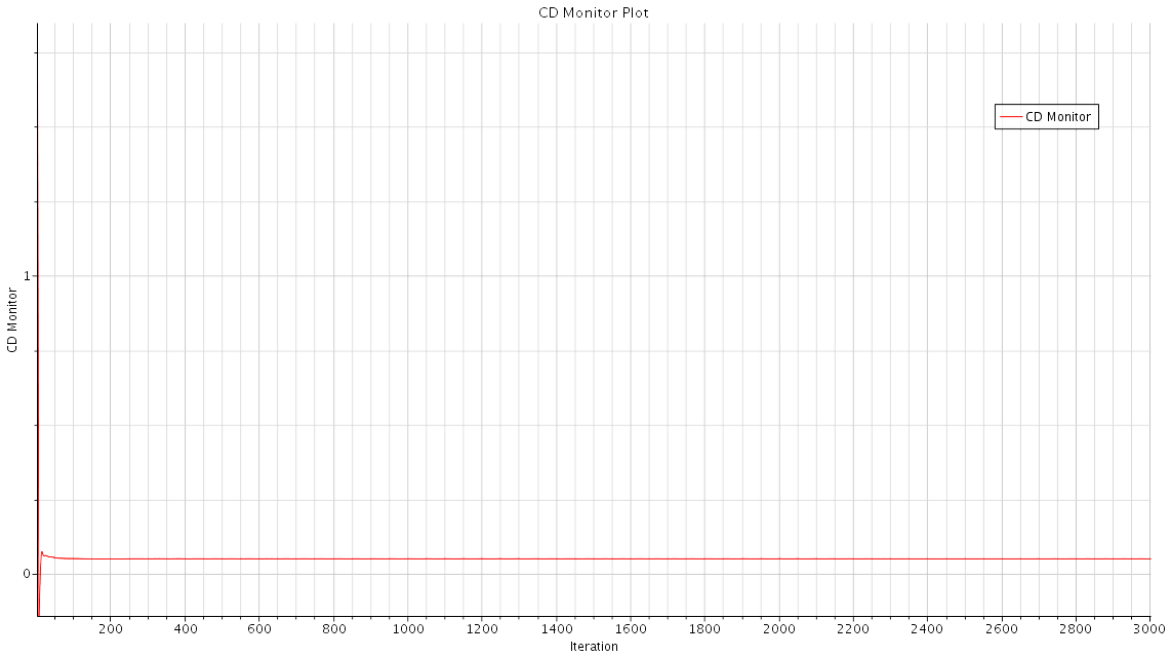


Figura 6.2: Convergenza del coefficiente di resistenza

ID		Net
15	Muso_DX	6,1%
16	Muso_SX	6,1%
5	Fusoliera_ant_DX	-0,9%
6	Fusoliera_ant_SX	-0,9%
7	Fusoliera_post_DX	1,4%
8	Fusoliera_post_SX	1,5%
3	Coda_DX	0,4%
4	Coda_SX	0,4%
1	Carrello_DX	4,5%
2	Carrello_SX	4,5%
21	Semiala_DX	20,9%
22	Semiala_SX	21,2%
25	Winglet_DX	0,5%
26	Winglet_SX	0,5%
13	Motore_DX	16,5%
14	Motore_SX	16,5%
9	Impennaggio_oriz_DX	-2,8%
10	Impennaggio_oriz_SX	-2,8%
23	Winglet_coda_DX	0,1%
24	Winglet_coda_SX	0,1%
17	Pinna_alta_DX	1,0%
18	Pinna_alta_SX	1,0%
19	Pinna_bassa_DX	0,2%
20	Pinna_bassa_SX	0,2%
11	Impennaggio_vert_DX	2,0%
12	Impennaggio_vert_SX	1,9%

Tabella 7: Distribuzione resistenze per PID

I valori percentuali della Tabella 7 sono da riferirsi alla resistenza totale del velivolo che, per ragioni di segreto industriale, non può essere riportato.

	Net
Fusoliera	25%
ALA	43%
Motori	33%
HTU	-5%
VTU	6%

Tabella 8: Distribuzioni resistenze per componenti generali del velivolo

Il valore di resistenza negativa per quanto riguarda la zona degli impennaggi orizzontali è dovuto al fatto che il velivolo è stato studiato nella configurazione non trimmata, quindi ad un angolo dell'equilibratore che non corrisponde a quello reale nelle condizioni di crociera.

Come risulta più chiaro dalla Figura 6.3, il punto di ristagno dell'impennaggio orizzontale risulta spostato verso il dorso. Ciò è dovuto alla variazione d'incidenza del flusso indotta dalla rotazione delle eliche, che, nella sezione di mezzeria del velivolo, danno un notevole contributo di velocità verso il basso.

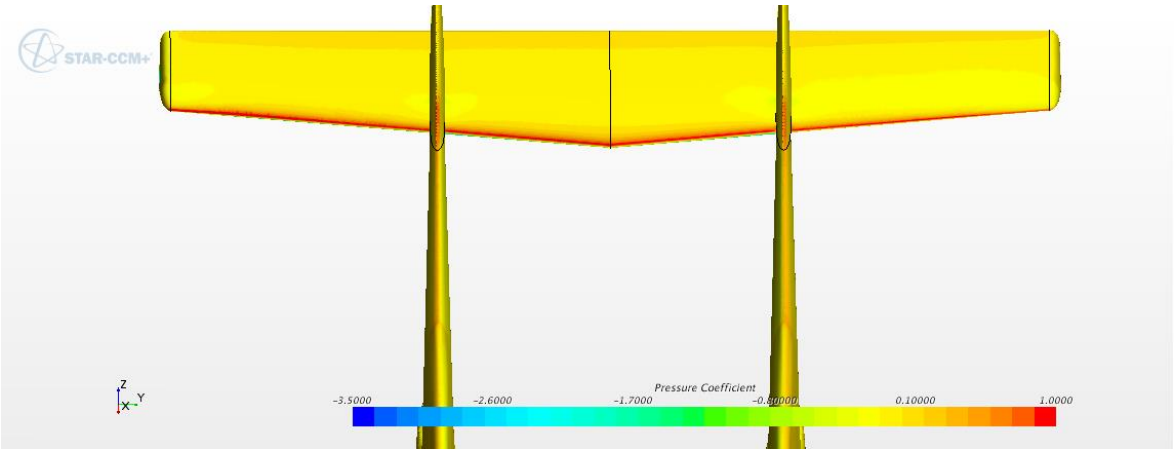


Figura 6.3: Visualizzazione del Cp sul bordo d'attacco dell'impennaggio orizzontale

In pratica il profilo simmetrico dell'impennaggio orizzontale viene investito da un flusso a incidenza negativa che genera portanza in direzione perpendicolare ad esso con una componente lungo l'asse x (del sistema assi vento, quindi a 3° di incidenza positiva) maggiore della componente della resistenza dell'impennaggio orizzontale lungo lo stesso asse.

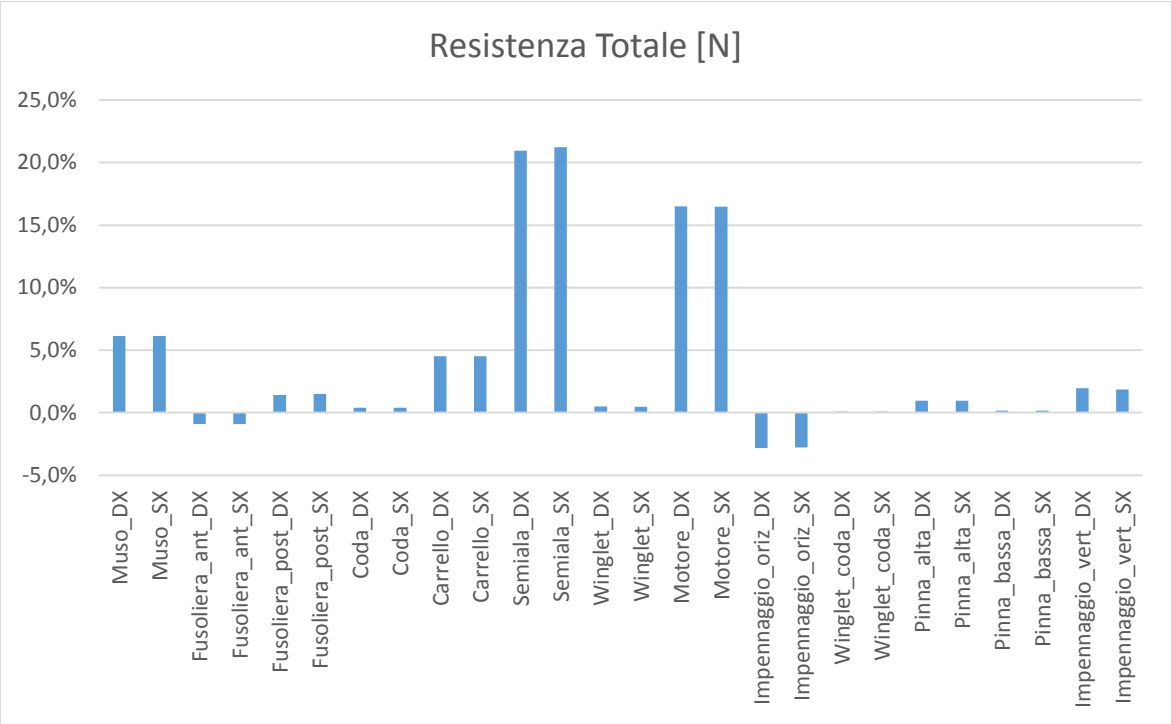


Figura 6.4: Istogramma distribuzione resistenze PID

Dall'istogramma di Figura 6.4 si deduce, come d'altronde ci si aspettava, che non v'è asimmetria del flusso. Infatti i valori della resistenza della suddivisione destra/sinistra sono pressoché uguali. Nell'istogramma seguente ho scisso in 2 il contributo della fusoliera isolando il contributo del carrello.

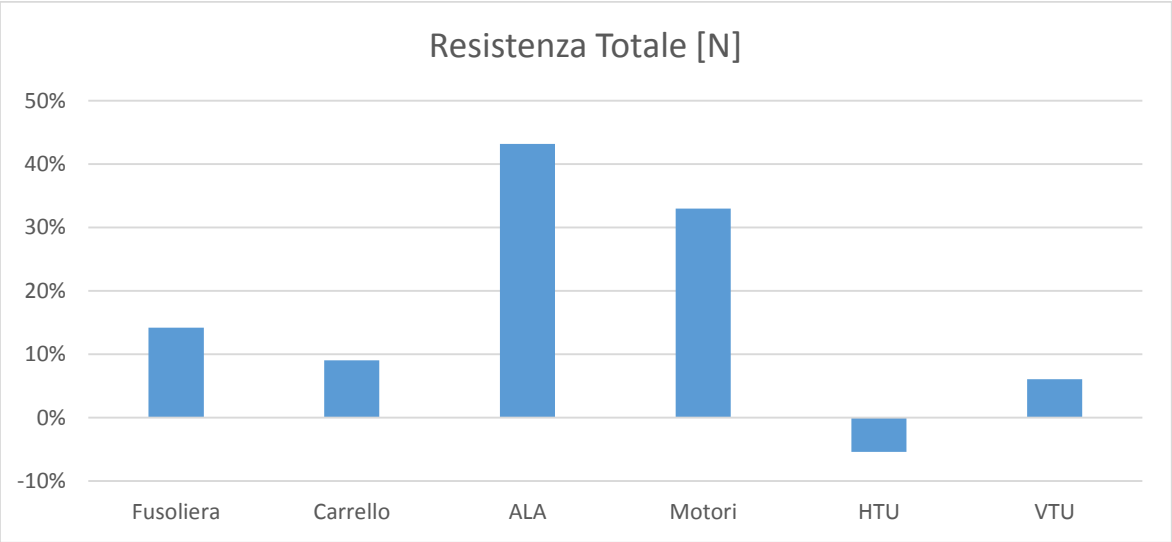


Figura 6.5: istogramma distribuzione di resistenza per componenti principali del velivolo

Come si vede bene dall'istogramma di Figura 6.5, i contributi maggiori alla resistenza complessiva (escludendo ala e fusoliera che non possono essere soggetti a modifiche consistenti) provengono dal carrello e dal motore. Per quanto riguarda il carrello si propone una soluzione retraibile fin dentro la fusoliera. Dalla Figura 6.6, che rappresenta gli sforzi tangenziali, è possibile comprendere la causa dell'aumento della resistenza indotta dalla presenza dei carrelli, infatti dietro la parte posteriore, il flusso è separato.

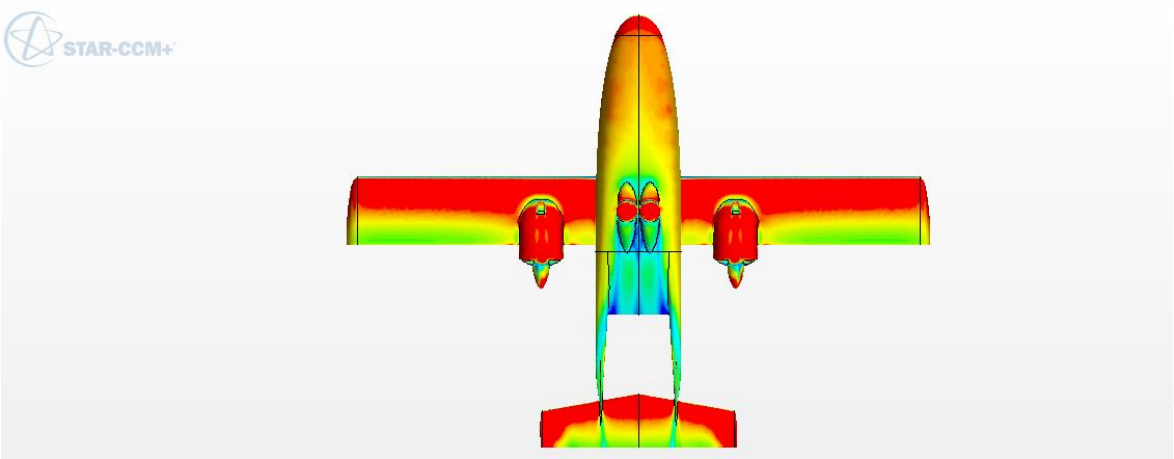
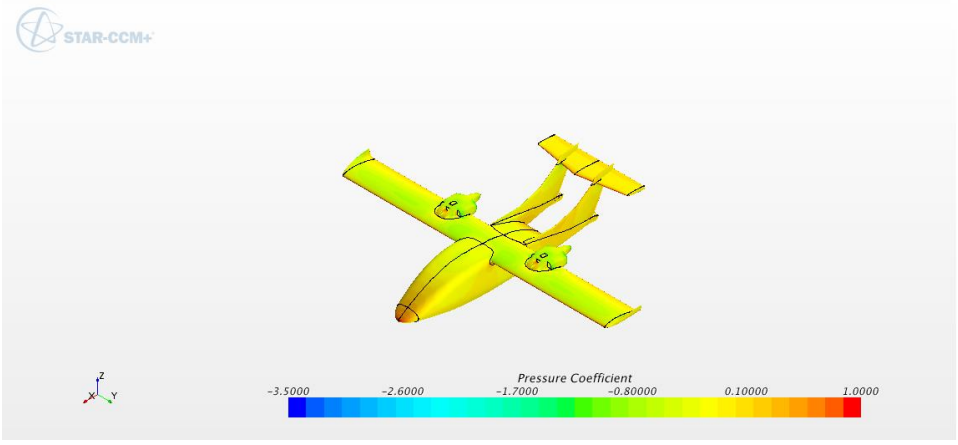
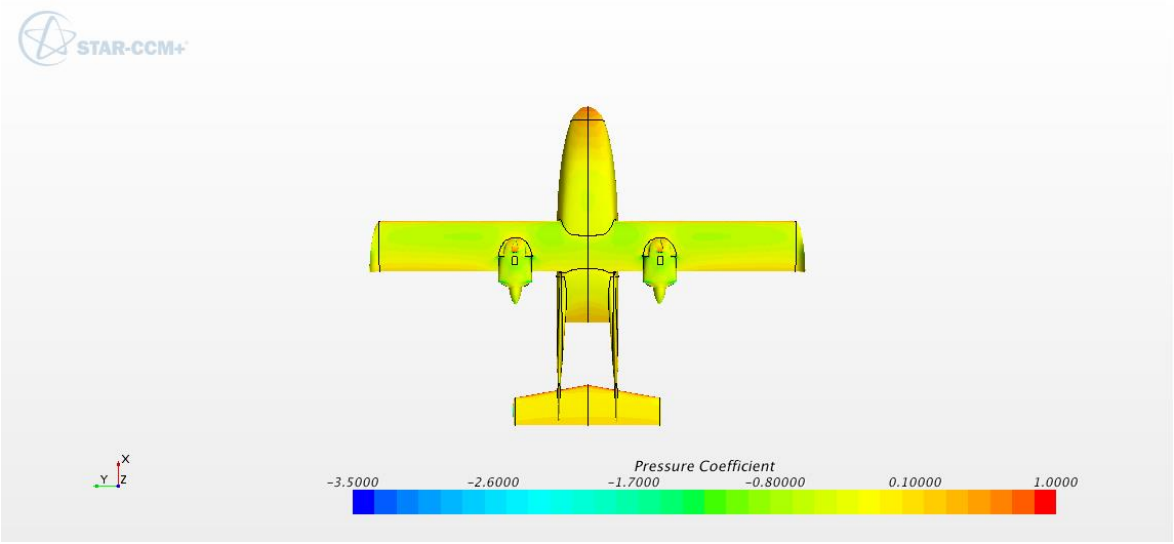


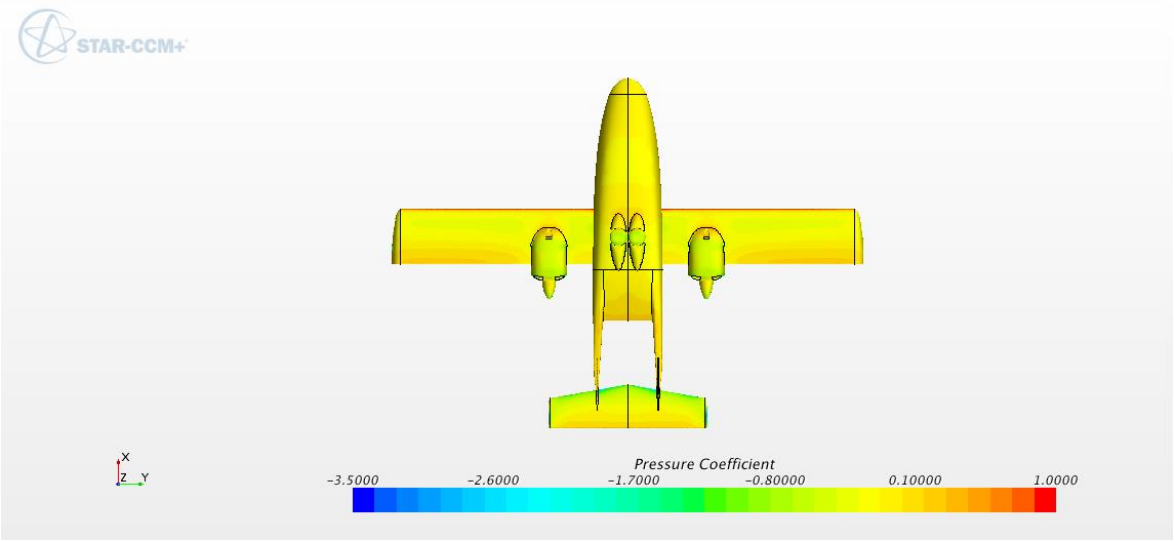
Figura 6.6: Sforzi tangenziali

Il motore invece ha richiesto un'analisi più dettagliata che vedremo nel prossimo paragrafo.

Verranno ora riportate le immagini dei coefficienti di pressione, degli sforzi tangenziali e delle linee di corrente per il velivolo analizzato:







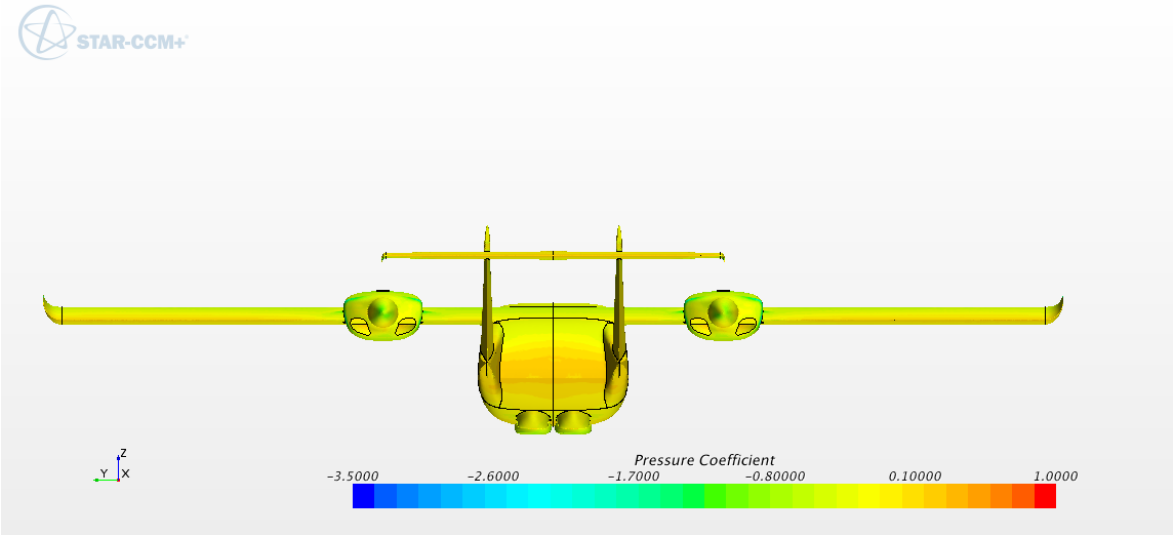


Figura 6.7: Cp velivolo

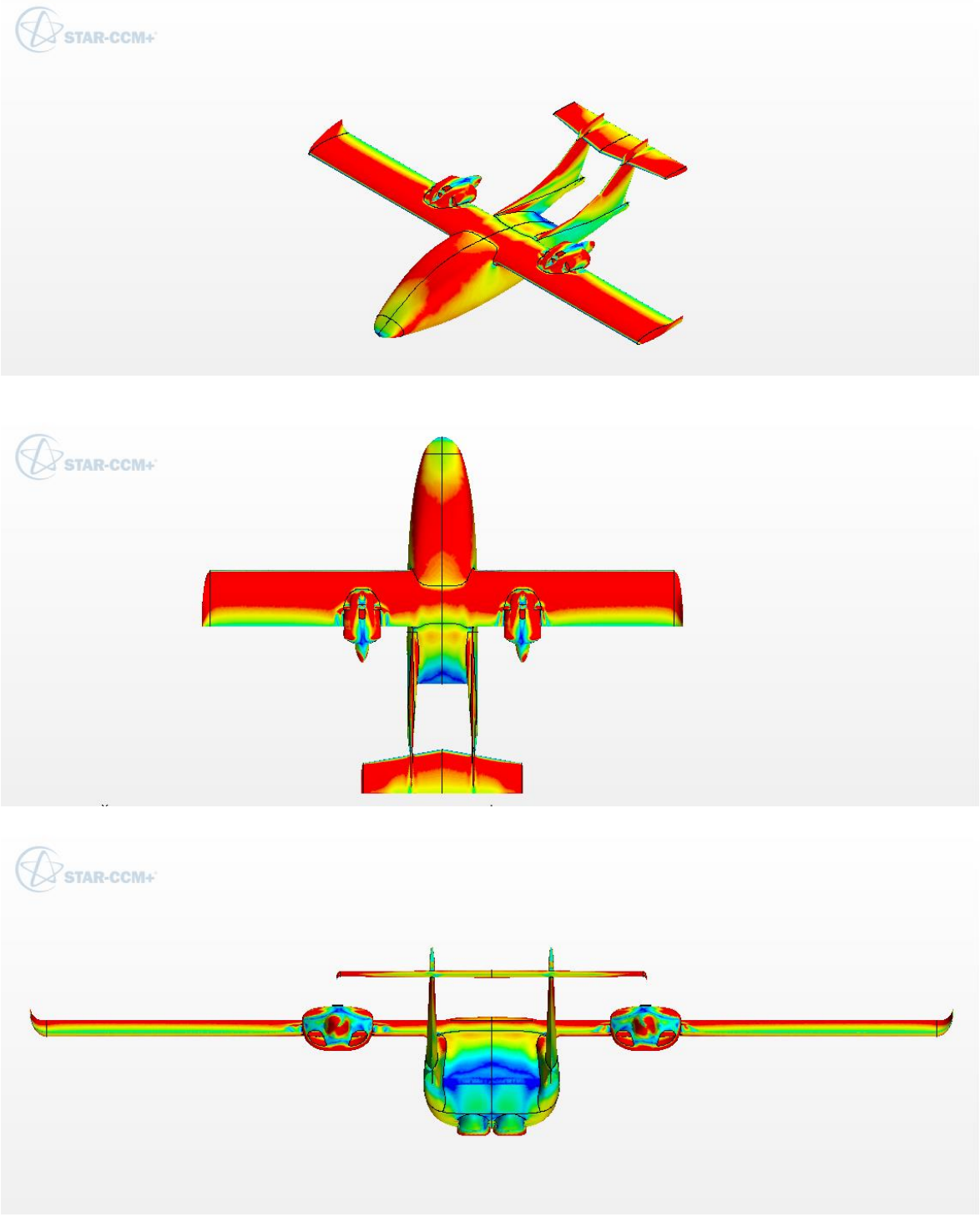


Figura 6.8: Wall Shear Stress

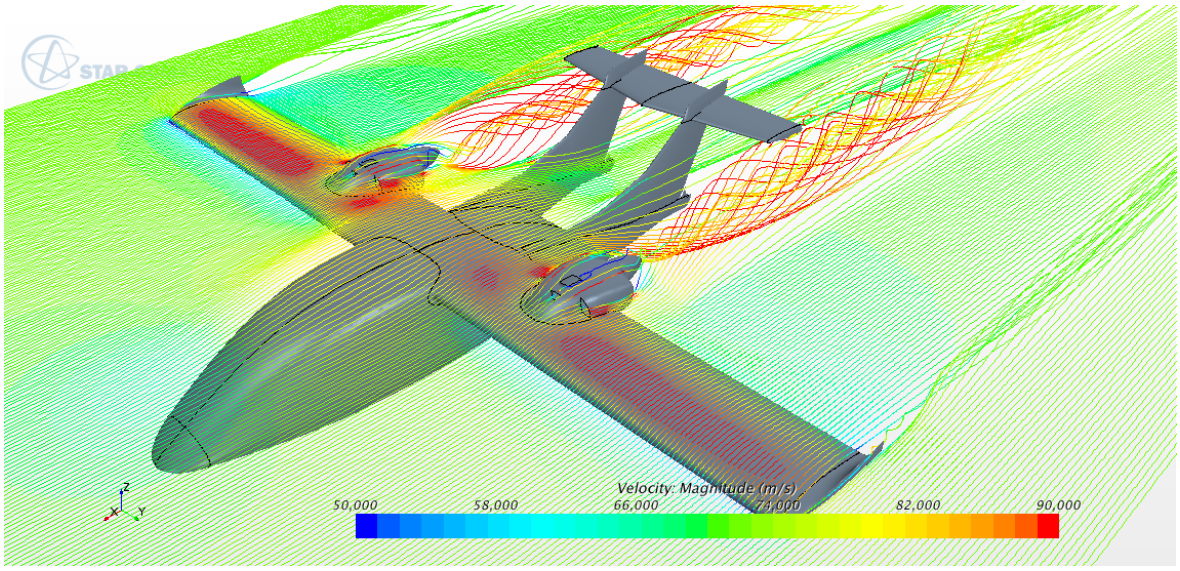


Figura 6.9: Linee di corrente

6.3 Analisi della gondola motore

E' stata fatta un'analisi CFD del motore isolato posto al centro dell'ala di uguale profilo di quella originale, ma di lunghezza pari all'intera apertura alare del velivolo. Questo per isolare quanto più è possibile il motore da effetti di interferenza indotti dall'ala quale il *tip vortex*. L'ideale sarebbe stato simulare un'ala infinita, che in termini di simulazione implica un'ala di apertura pari alla larghezza del box di calcolo, ma richiede un costo computazionale decisamente troppo alto. Così sono state fatte varie prove di convergenza con domini di diverse dimensioni finché non si sono trovate le dimensioni minime che garantiscano la convergenza. Il dominio è rappresentato nella Figura 6.10

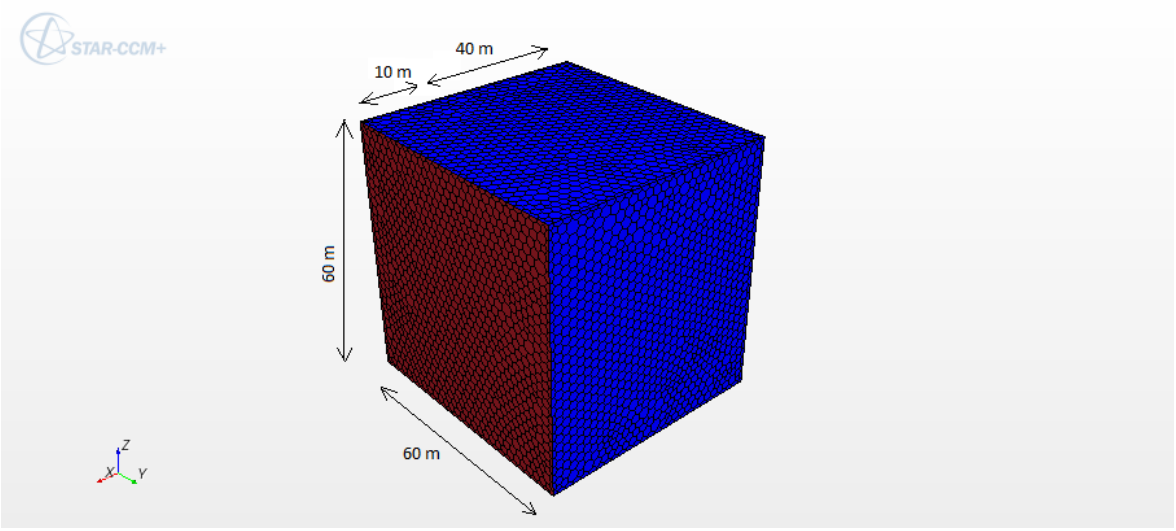


Figura 6.10: Dominio motore isolato

Il dominio scelto è:

- 50 m di lunghezza di cui 10 m a monte del motore e 40 m a valle;
- 60 m di larghezza;
- 60 m di altezza;

Inoltre, per capire al meglio la causa dell'aumento di resistenza del motore, è stato fatto uno split in due, separando la parte anteriore da quella posteriore.

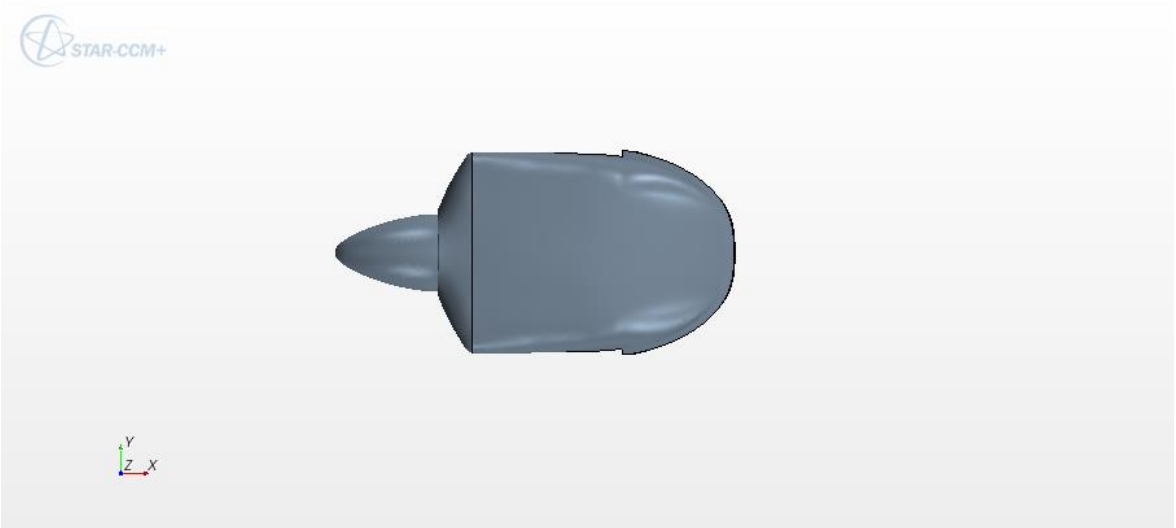


Figura 6.11: Split motore

L’idea base è che, nella parte posteriore del motore, a causa non solo della particolare forma della calotta di chiusura, ma anche per l’effetto dell’aspirazione dovuto all’elica, le linee di corrente separino facendo sì che la scia abbia uno spessore maggiore. Inoltre, collegato a quest’ultimo discorso, ci si aspetta anche un elevato valore della resistenza di pressione.

I settaggi della simulazione sono gli stessi che si sono usati per la configurazione del velivolo completo (quindi considerando anche l’effetto dell’elica). L’unica differenza risiede nelle zone di ingresso/uscita di portata d’aria che sono state chiuse per questioni di semplicità del modello (vedi Figura 6.11). D’altronde, a seguito di numerose simulazioni, si è constatato che l’effetto delle condizioni di *Mass Flow Inlet* e *Flow Split Outlet*, per il caso in esame, è trascurabile. Di seguito viene riportato il grafico dei residui per la corrente simulazione:

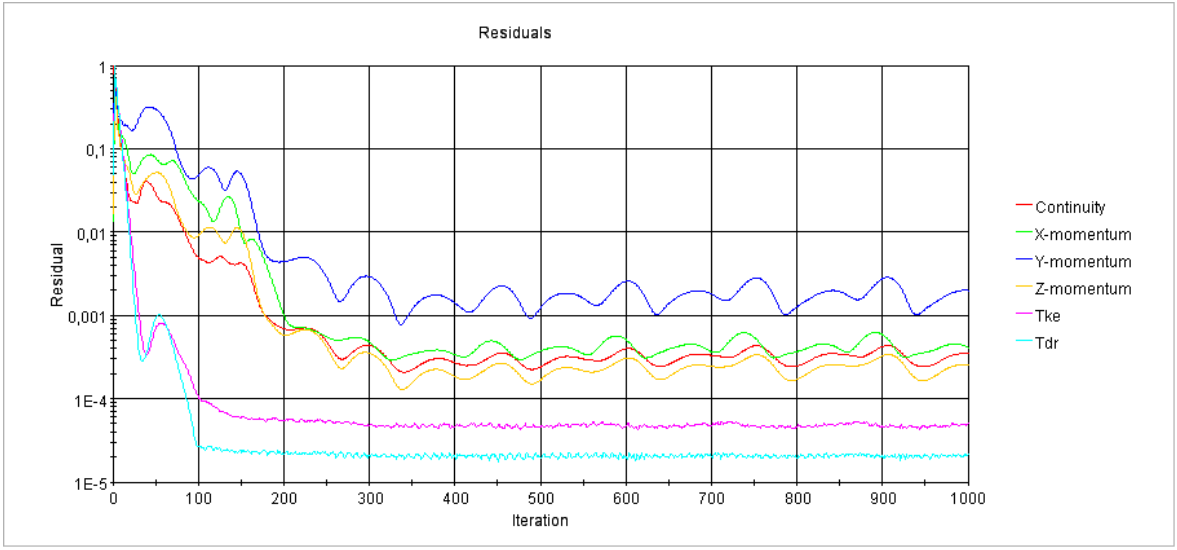


Figura 6.12: Residui motore originale

Data la semplicità del modello geometrico, sono state effettuate solo 1000 iterazioni. Il valor medio così raggiunto è molto vicino (con un errore inferiore all’1%) a quello calcolato a partire dal velivolo intero. Come si vede bene dal grafico dei residui, l’andamento a convergenza presenta ancora delle fluttuazioni. Una spiegazione a tale fenomeno può ricondursi o al modello dell’elica che introduce oltre al salto di pressione anche l’andamento delle forze tangenziali e quindi la componente di velocità angolare nel flusso a valle dell’elica, o alla possibile instabilità del flusso che si genera nella parte compresa fra motore posteriore e elica. Per analizzare in modo più approfondito la causa delle fluttuazioni, sono state svolte delle visualizzazioni del

coefficiente di pressione lungo la superficie del motore. Si riporta in Figura 6.13 la visualizzazione del Cp ottenuto sul piano trasversale di mezzeria del motore:

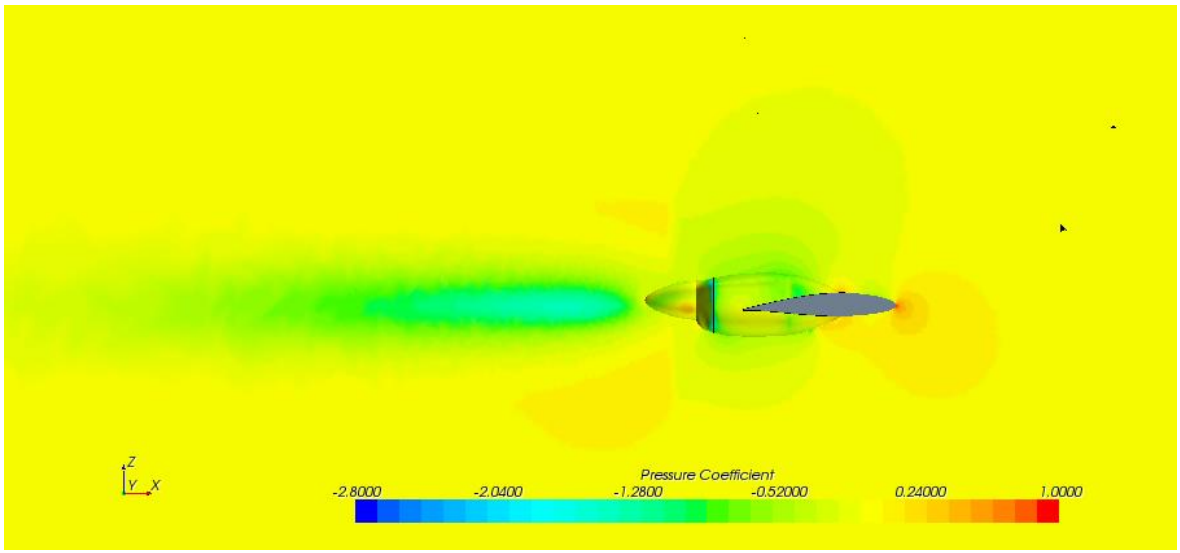


Figura 6.13: Cp piano di mezzeria del motore

Come si vede, nella zona a confine fra parte anteriore e parte posteriore del motore vi è un'elevata aspirazione, dovuta all'eccessiva curvatura del bordo della calotta di chiusura della gondola motore, che provoca separazione del flusso. A conferma della separazione, si riporta in Figura 6.14 anche la visualizzazione della distribuzione delle forze tangenziali alla superficie. Le zone in rosso indicano flusso attaccato, mentre le zone blu indicano flusso separato. Le zone a fasce di colore intermedio indicano un ispessimento dello strato limite.

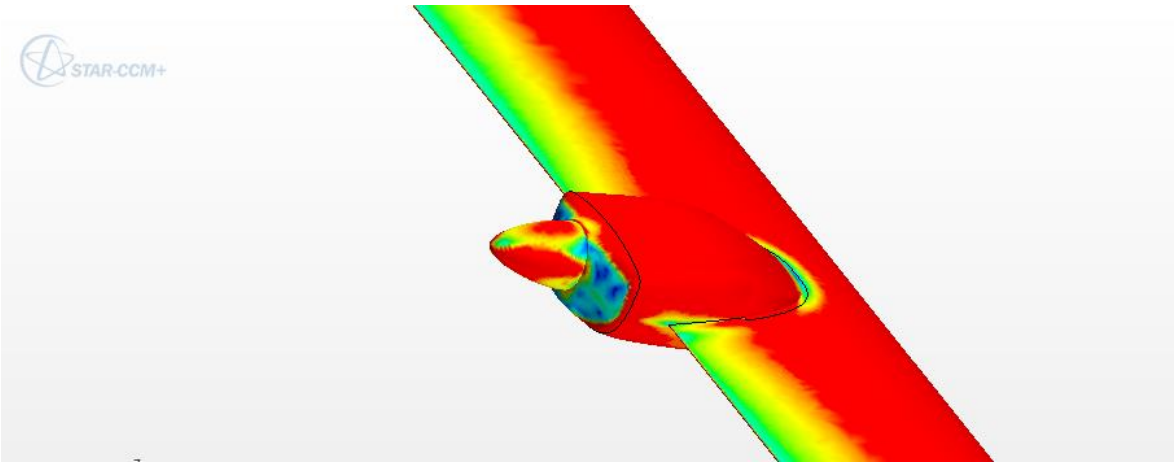


Figura 6.14: Wall Shear Stress motore "originale"

Infine è stata fatta anche una visualizzazione del campo di vorticità assiale lungo il piano trasversale situato nella zona fra motore posteriore ed elica.

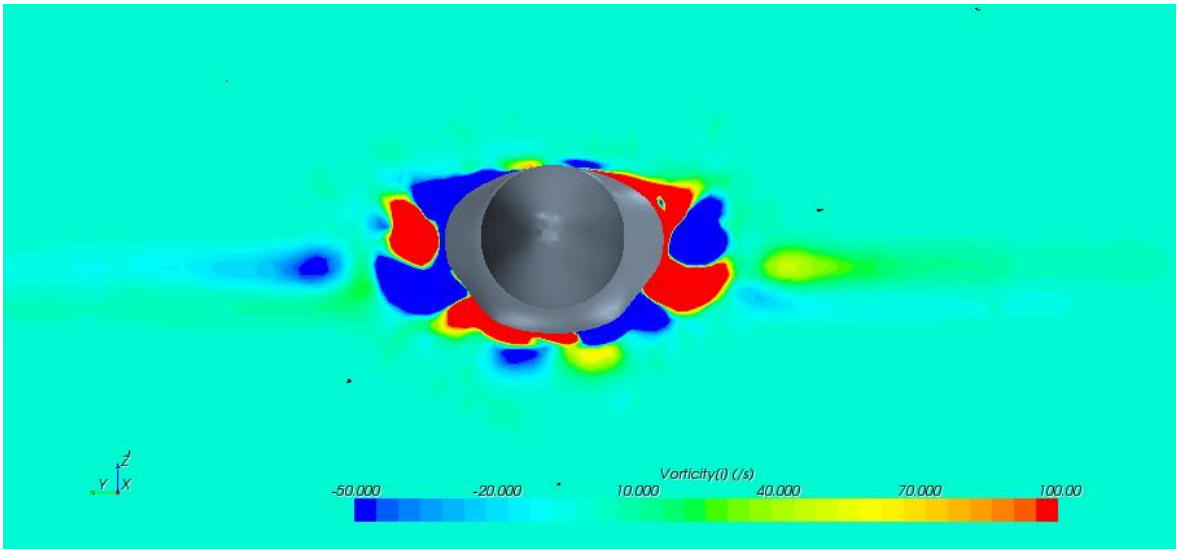


Figura 6.15: Vorticità assiale motore originale

Effettivamente la parte posteriore del motore è riconducibile più ad un corpo tozzo che aerodinamico, quindi la presenza di tali vortici non sorprende. Avendo però svolto un'analisi stazionaria, chiaramente il solutore presenta delle difficoltà ad arrivare a convergenza.

6.4 Modifica della gondola motore

Per cercare di ridurre la resistenza di pressione dovuta alla forma della parte posteriore del motore, si è proposta una modifica. Spostando il motore in avanti verso il bordo d'attacco, si può dare una forma più "aerodinamica" alla calotta di chiusura del vano motore. Questa soluzione chiaramente è solo rappresentativa, per così dire, nel senso che non è stata effettuata una progettazione vera e propria, ma è comunque interessante per capire effettivamente quanto potrebbe convenire (in termini di riduzione della resistenza) operare delle modifiche di questa tipologia. Nella Figura 6.16 viene riportata l'immagine della nuova configurazione geometrica del motore che si andrà ad analizzare:

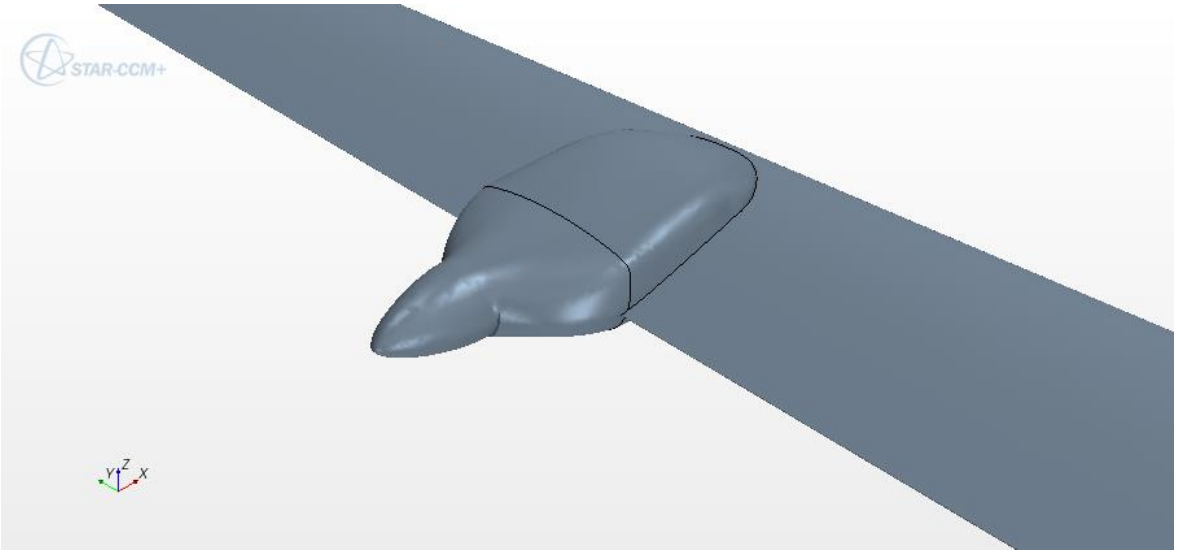


Figura 6.16: Modifica della calotta di chiusura del motore

Per assicurare la validità dei risultati rispetto alla precedente simulazione, sono stati imposti gli stessi settaggi, le stesse dimensioni del dominio e la stessa semplificazione per quanto riguarda le prese d'aria del motore. Di seguito viene riportato l'andamento dei residui:

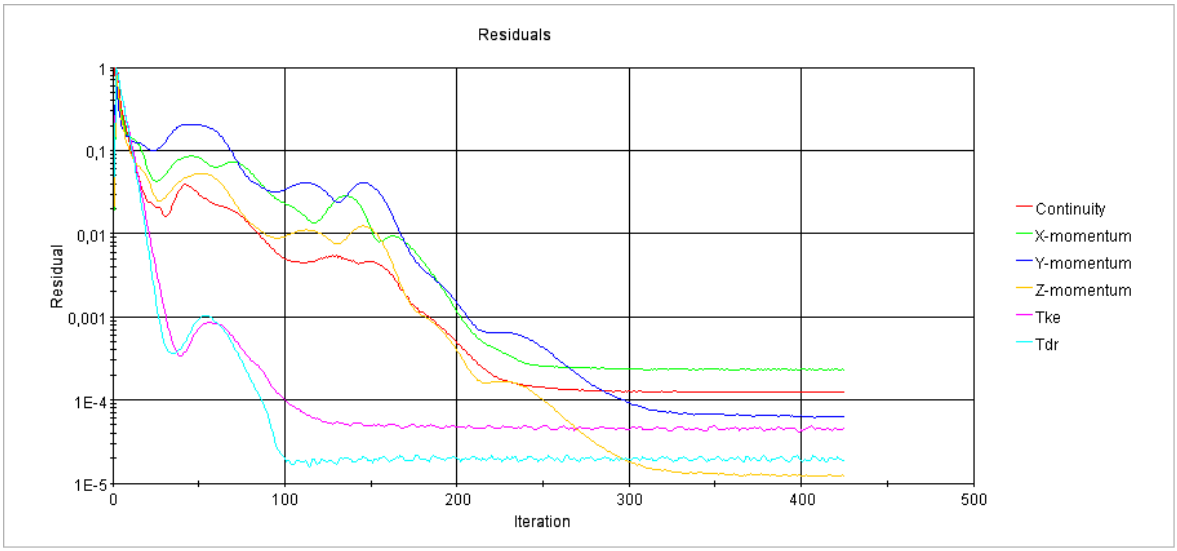


Figura 6.17: Residui motore modificato

Come si vede, l'analisi è andata a convergenza e addirittura con un numero di iterazioni inferiore, sintomo che le non stazionarietà del flusso non sono più presenti. Infatti si è ottenuta una riduzione del 70% rispetto al valore originale. Come già detto, questa riduzione è stata ottenuta mediante una modifica geometrica della calotta di chiusura, nonché della posizione sull'ala del motore, che non è stata oggetto di una vera e propria progettazione. Potremmo considerare questo caso come situazione di ottimo (almeno dal punto di vista della riduzione della resistenza aerodinamica), ma sicuramente vi è un ampio margine di miglioramento e operare una modifica strutturale in tal senso potrebbe comunque consentire un sostanziale abbattimento della resistenza complessiva del velivolo.

Di seguito verranno riportato le immagini dei Cp, degli sforzi tangenziali e del campo di vorticità esattamente come nel caso precedente per completare il confronto:

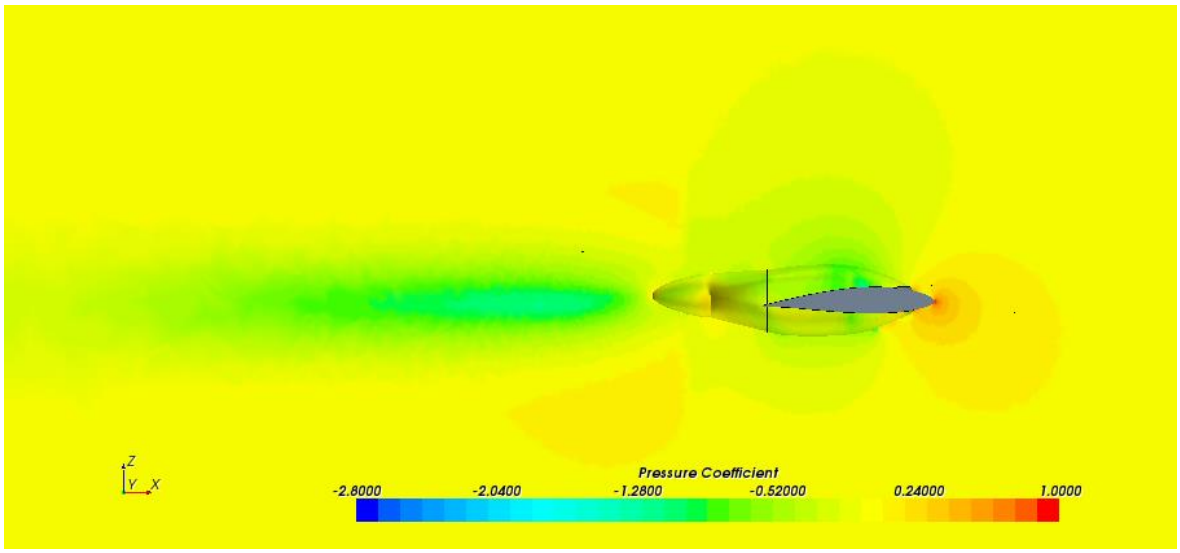


Figura 6.18: Cp piano di mezzeria del motore nuovo

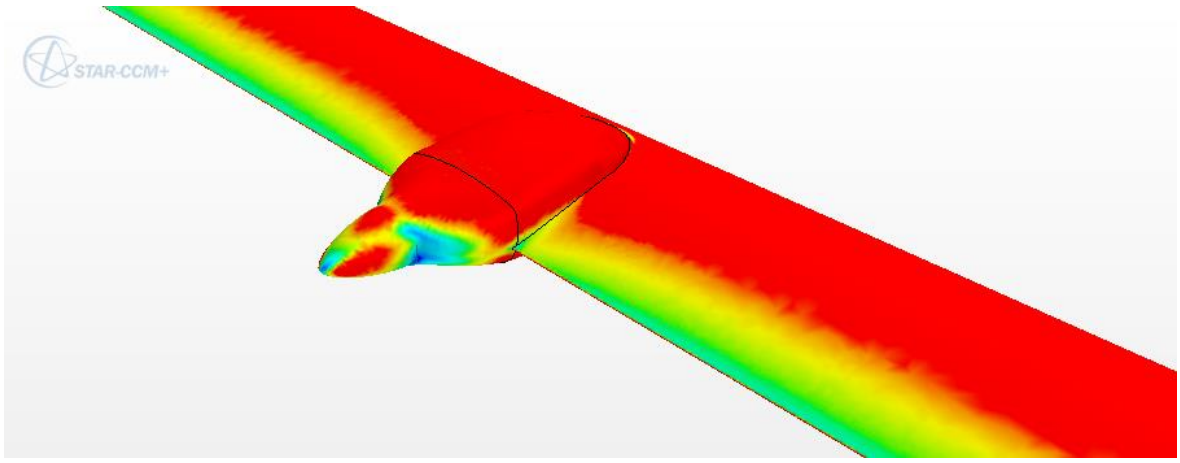


Figura 6.19: Wall Shear Stress motore nuovo

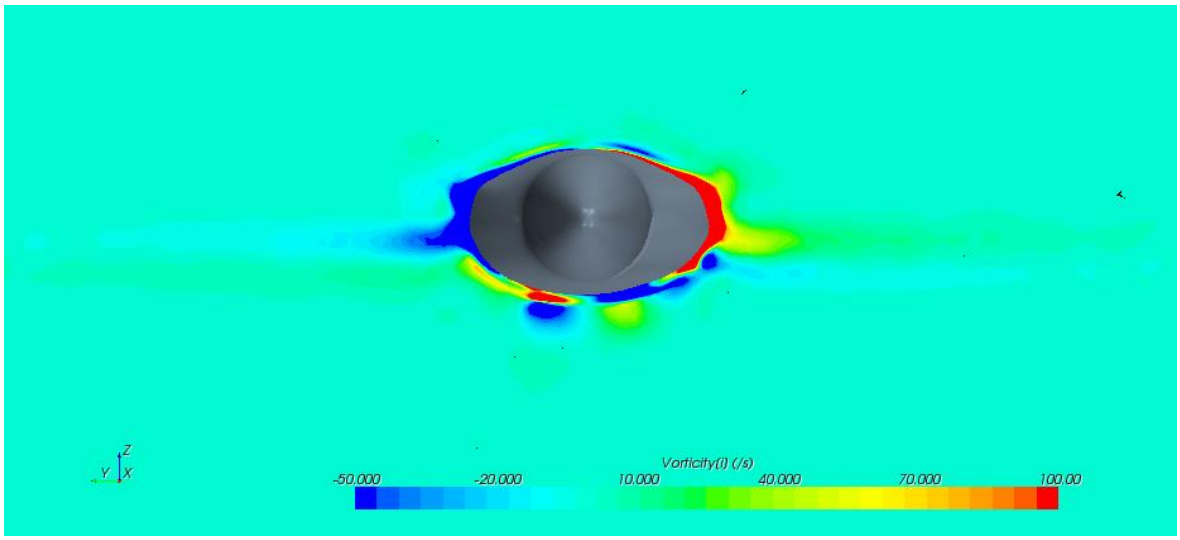


Figura 6.20: Vorticità assiale motore nuovo

Da notare, nell'immagine del coefficiente di pressione, la mancanza della zona d'aspirazione nella linea di confine fra parte posteriore e anteriore. Infatti, come si vede bene dall'immagine degli sforzi tangenziali, il flusso separa più a valle e solo per l'effetto dell'elica. Da queste semplici considerazioni emerge la conseguente riduzione della formazione e dell'intensità dei vortici nonché il minor spessore della scia.

Sulla base di questi risultati, sono state svolte altre 3 simulazioni sul velivolo completo:

1. Velivolo completo con motore originale e senza carrelli
2. Velivolo completo con motore nuovo e carrelli
3. Velivolo completo con motore nuovo e senza carrelli

Nei prossimi paragrafi verranno singolarmente presentati e analizzati soffermandoci soprattutto sull'eventuale guadagno in termini di resistenza complessiva del velivolo.

6.5 Caso 1

Nel caso 1 è stata fatta la simulazione CFD del velivolo con l'unica modifica del reparto carrelli che sono stati completamente tolti perché rientranti in fusoliera. I settaggi del simulatore sono chiaramente immutati. Di seguito vengono riportati i grafici di convergenza:

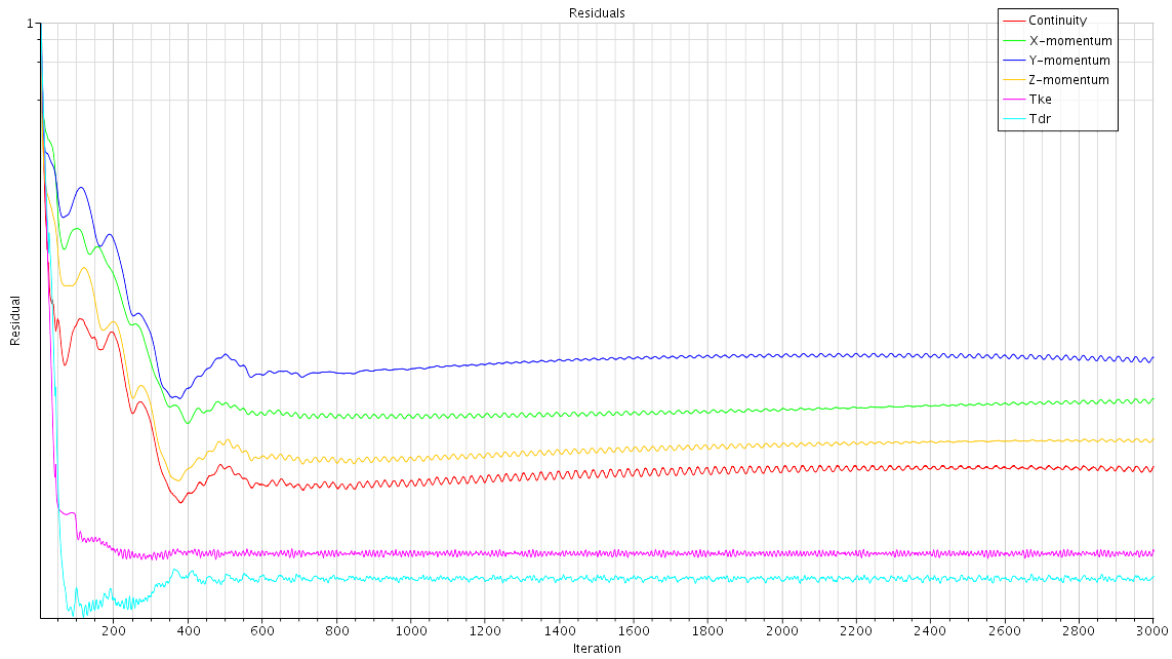


Figura 6.21: Residui caso 1

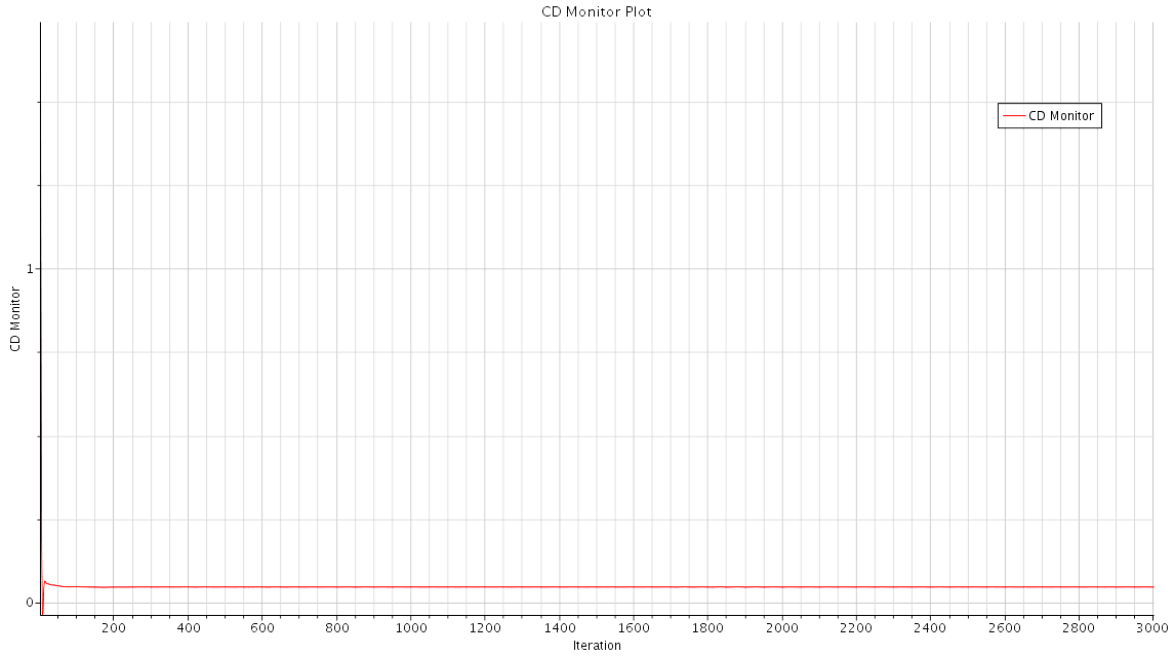


Figura 6.22: Convergenza Cd caso 1

Il guadagno di resistenza ottenuto per il caso 1 è riportato in Tabella 9

	%
Fusoliera	11%
ALA	0,3%
Motori	-0,1%
HTU	0,1%
VTU	-0,5%

Tabella 9: Guadagno percentuale caso 1

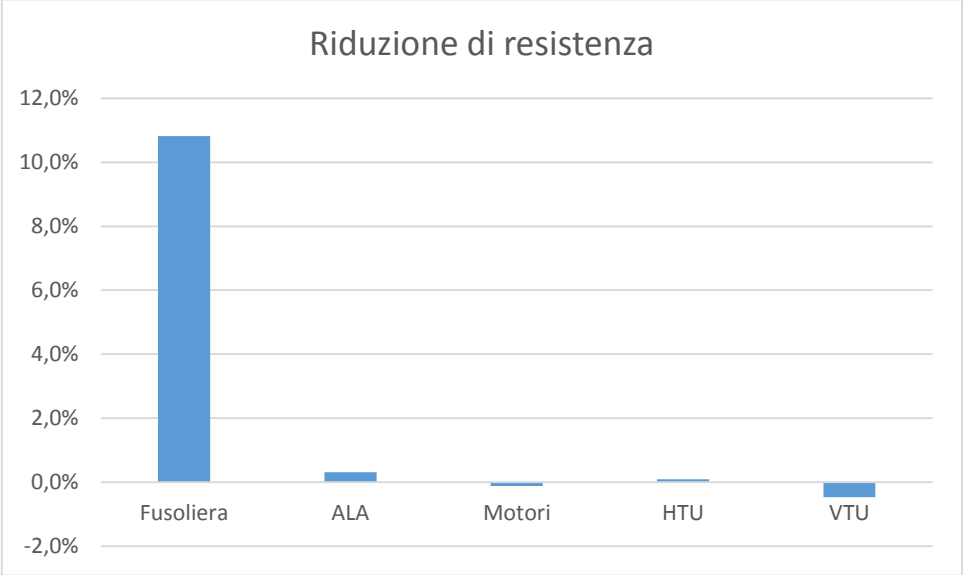


Figura 6.23: Istogramma riduzione di resistenza caso 1

I valori percentuali sono da riferirsi alla resistenza totale del velivolo senza alcuna modifica apportata. D'ora in poi tale valore sarà preso di riferimento per tutti gli altri casi.

	delta (N)
Muso_DX	-1,72
Muso_SX	-1,75
Fusoliera_ant_DX	-7,38
Fusoliera_ant_SX	-7,29
Fusoliera_post_DX	-3,47
Fusoliera_post_SX	-5,11
Coda_DX	1,33
Coda_SX	1,47
semiala_DX	0,83
semiala_SX	0,82
Winglet_DX	0,00
Winglet_SX	0,00
Motore_DX	-3,89
Motore_SX	-3,98
Impennaggio_oriz_DX	1,23
Impennaggio_oriz_SX	1,12
Winglet_coda_DX	-0,04
Winglet_coda_SX	-0,32
Pinna_alta_DX	0,06
Pinna_alta_SX	0,20
Pinna_bassa_DX	-0,08
Pinna_bassa_SX	-0,10
Impennaggio_vert_DX	5,16
Impennaggio_vert_SX	6,23

Tabella 10: Differenze resistenza per PID caso 1

E' stato omesso volutamente la differenza di resistenza ottenuta per la mancanza del carrello. La Tabella 10 è stata comunque riportata per validare il confronto con la simulazione di riferimento. Come si vede chiaramente, le differenza dei valori di resistenza con il caso di riferimento sono veramente irrisorie (sono espresse in N).

La riduzione di resistenza ottenuta per il caso 1 è del 10,8% rispetto al caso iniziale. Verranno ora riportate le immagini dei coefficienti di pressione, degli sforzi tangenziali e la visualizzazione delle linee di corrente:

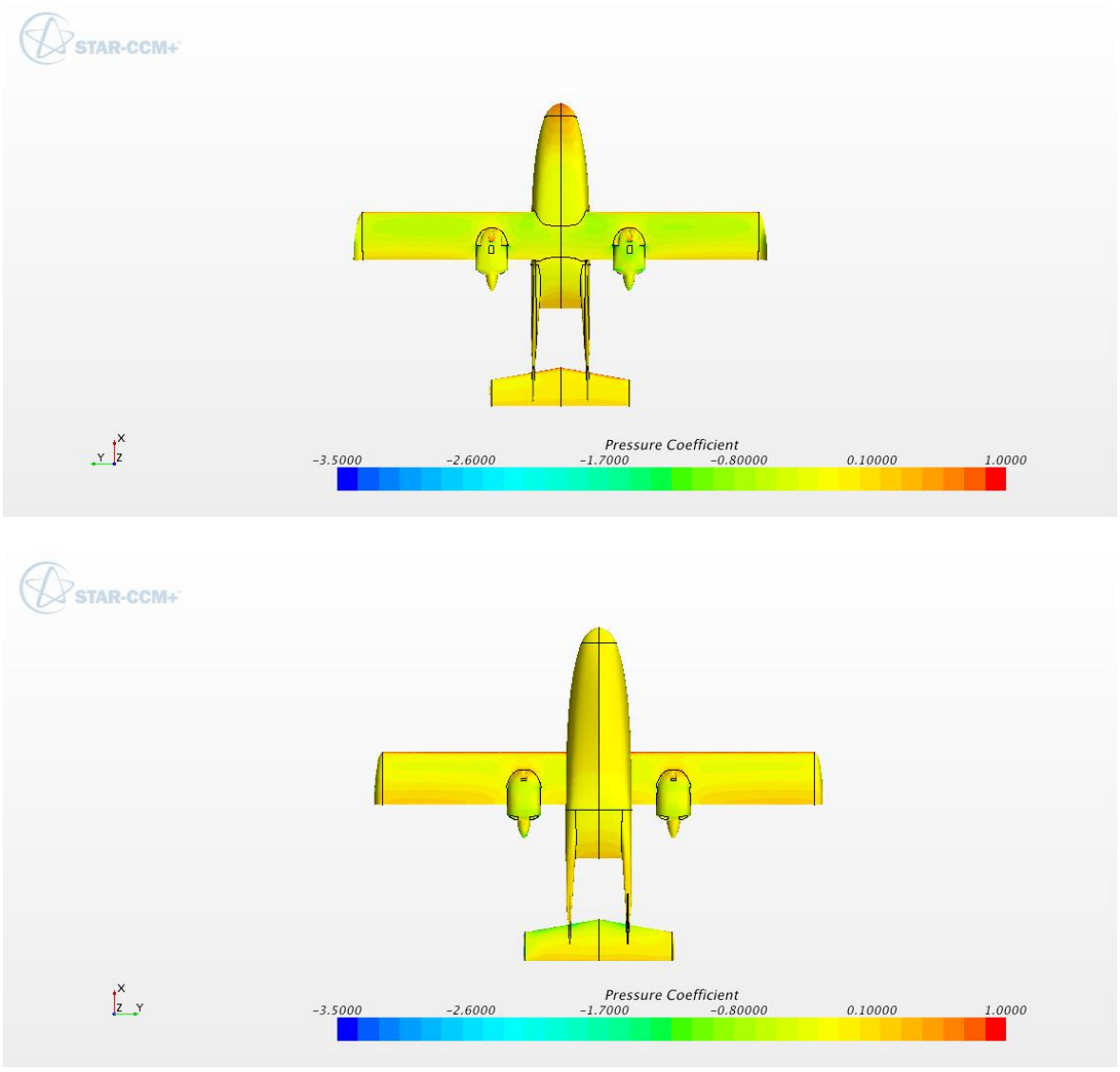


Figura 6.24: Cp caso 1

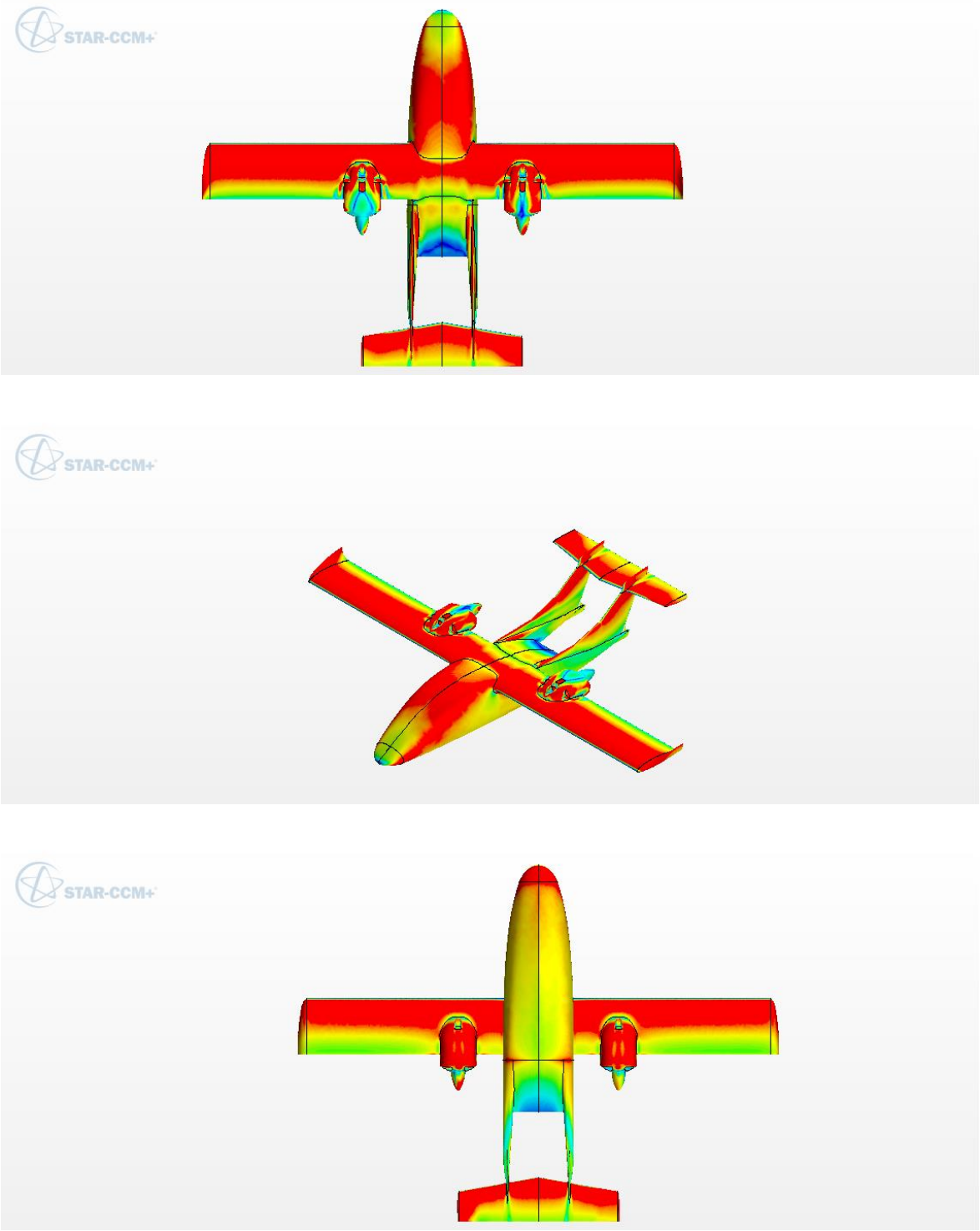


Figura 6.25: Wall Shear Stress caso 1

Notare la riduzione della separazione del flusso per l'assenza dei carrelli

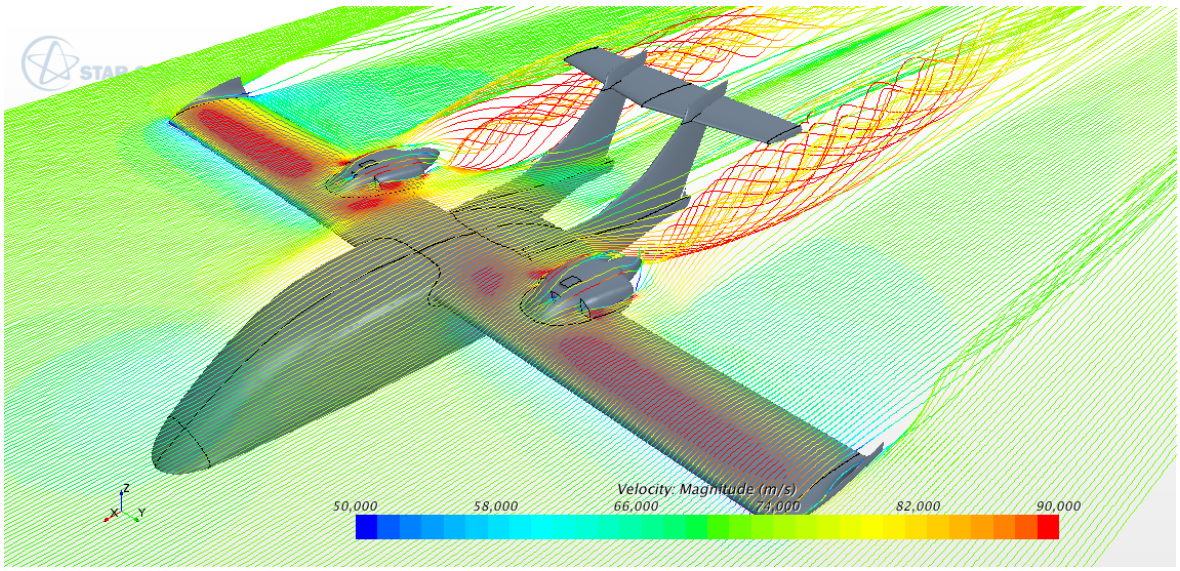


Figura 6.26: Linee di corrente caso 1

6.6 Caso 2

Nel caso 2 è stato simulato il velivolo completo (con carrelli) e la nuova configurazione del motore già analizzata nel paragrafo 6.4, ma senza alcuna semplificazione, quindi annettendo anche le prese d'aria.

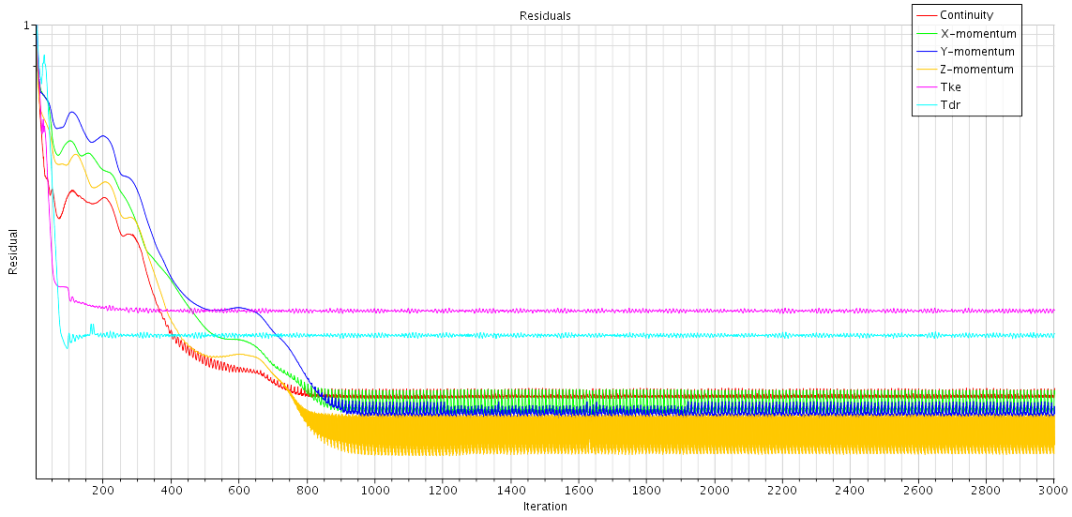


Figura 6.27: Residui caso 2

Da notare che i residui dell'equazioni di bilancio di quantità di moto sono dell'ordine di 10^{-7}.

Di seguito viene riportato il confronto con il caso iniziale e il conseguente guadagno in termini di resistenza percentuale

	Δ (N)
Muso_DX	-1,16
Muso_SX	-1,16
Fusoliera_ant_DX	-0,27
Fusoliera_ant_SX	0,12
Fusoliera_post_DX	2,87
Fusoliera_post_SX	2,61
Coda_DX	2,94
Coda_SX	3,19
Carrello_DX	1,55
Carrello_SX	0,71
semiala_DX	4,94
semiala_SX	4,21
Winglet_DX	9,93
Winglet_SX	10,11
Motore_DX	-196,12
Motore_SX	-200,29
Impennaggio_oriz_DX	0,98
Impennaggio_oriz_SX	1,21
Winglet_coda_DX	-3,97
Winglet_coda_SX	-3,99
Pinna_alta_DX	-0,94
Pinna_alta_SX	-0,76
Pinna_bassa_DX	0,03
Pinna_bassa_SX	0,04
Impennaggio_vert_DX	2,18
Impennaggio_vert_SX	2,19

Figura 6.28: Differenze resistenza per PID caso 2

Anche qui, a parte la sezione motori che è quella dove è stata apportata la modifica, i valori sono pressoché uguali a quelli del caso iniziale.

	%
Fusoliera	-0,6%
ALA	-0,1%
Motori	15,9%
HTU	0,4%
VTU	-0,2%

Tabella 11: Guadagno percentuale resistenza caso 2

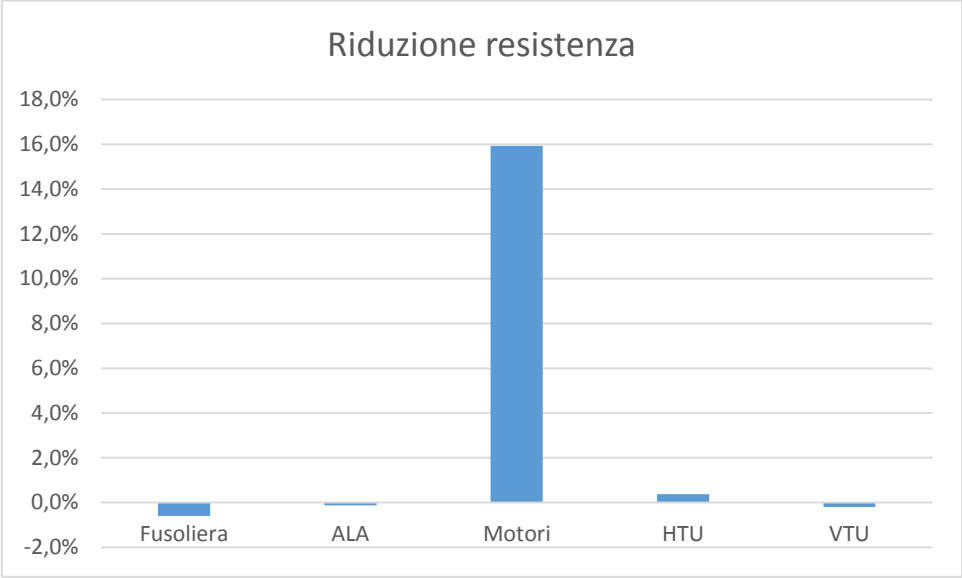


Figura 6.29: Istogramma riduzione di resistenza caso 2

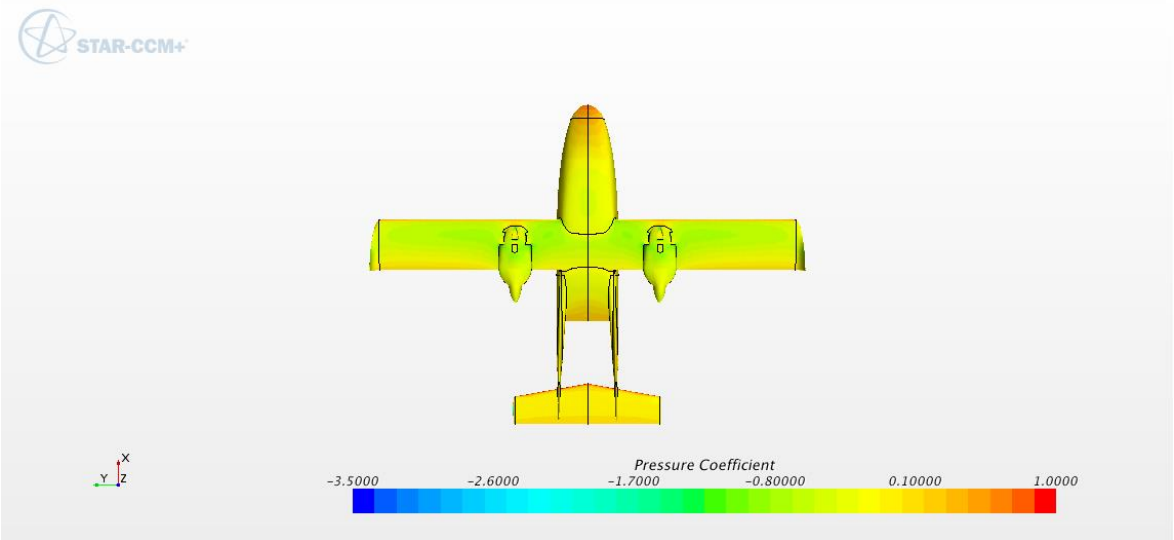
La riduzione di resistenza ottenuta per il caso 2 è del 15,4% rispetto al caso iniziale. Poiché il caso 2 è riferito alla nuova configurazione del motore, più spostato in avanti verso il bordo d'attacco dell'ala, è opportuno fare anche un confronto sull'eventuale perdita di portanza. Come già specificato più volte, i valori dei coefficienti di portanza del velivolo non sono stati riportati perché non rientrano nello scopo della tesi, ma sono stati comunque calcolati ed è sempre possibile fare il confronto fra due questo caso e quello di riferimento:

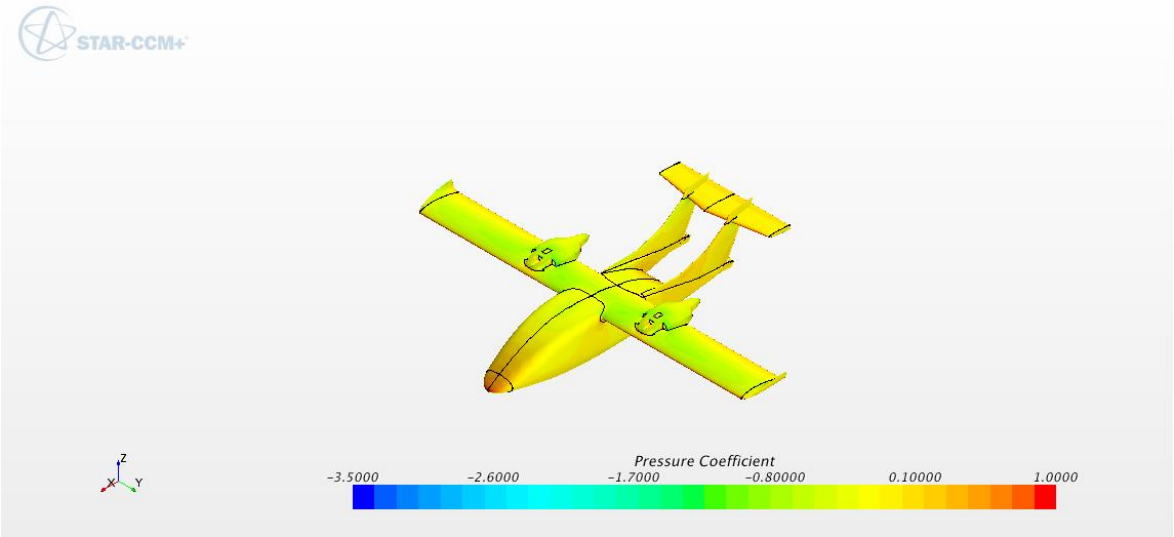
	ΔCl
Ala	0,004
Motori	0,0105
Totale	0,0145

Il totale rappresenta il 3,5% di differenza rispetto al Cl complessivo valutato dall'equazione di equilibrio con il peso in condizione di volo livellato. Inoltre il ΔCl è stato ricavato come differenza del caso 2 meno il caso originale di riferimento, sembra quindi che ci sia un piccolo aumento del coefficiente di portanza complessivo, ma questo potrebbe rientrare in un errore di

soluzione numerica tra l'altro al di sotto della soglia di tolleranza (fissata al 5%) e perciò trascurabile.

Verranno ora riportate le immagini dei coefficienti di pressione, degli sforzi tangenziali e la visualizzazione delle linee di corrente:





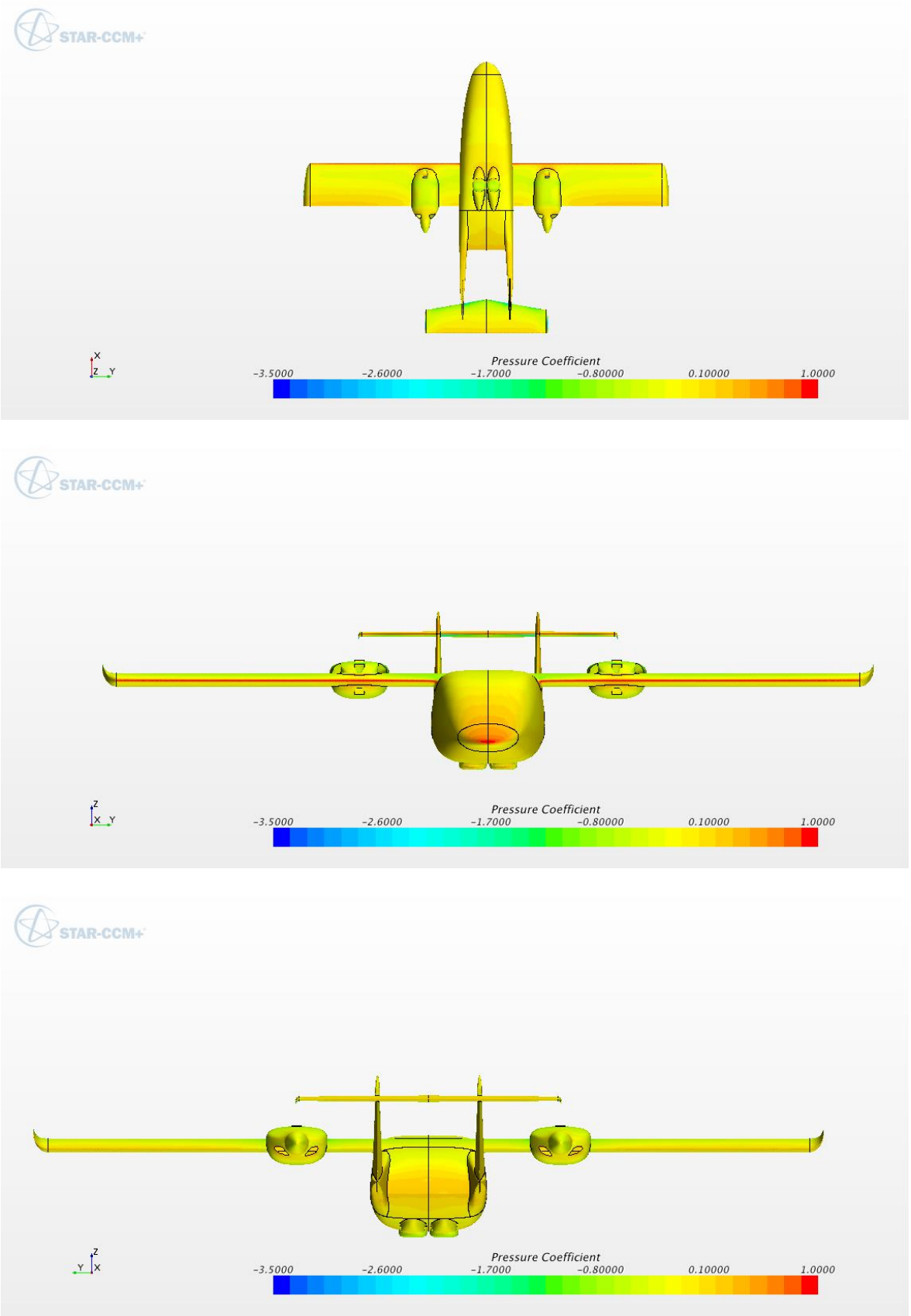
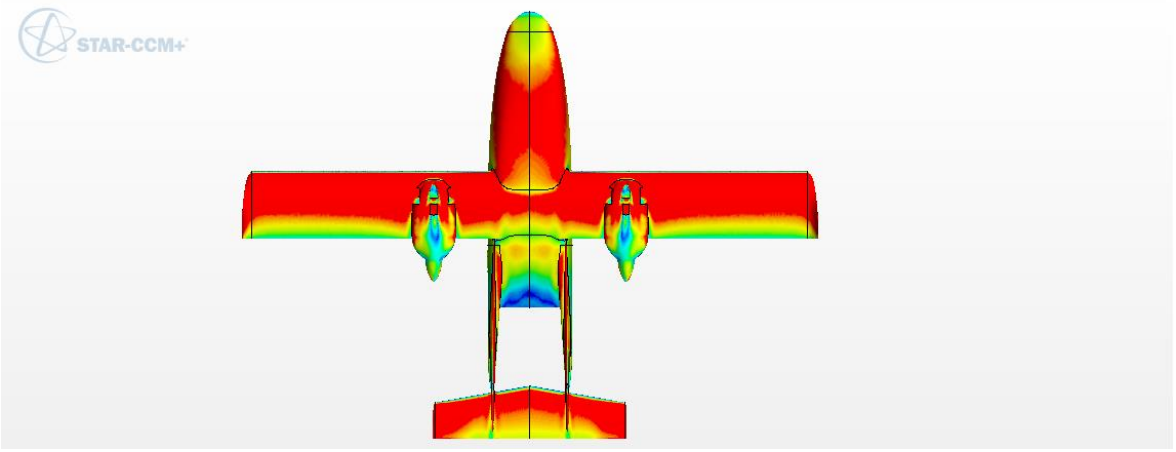
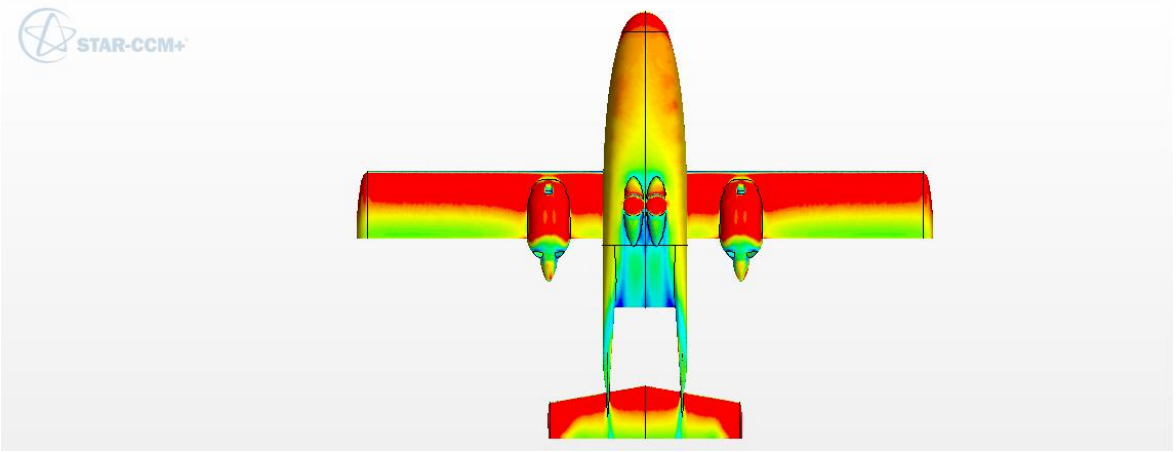
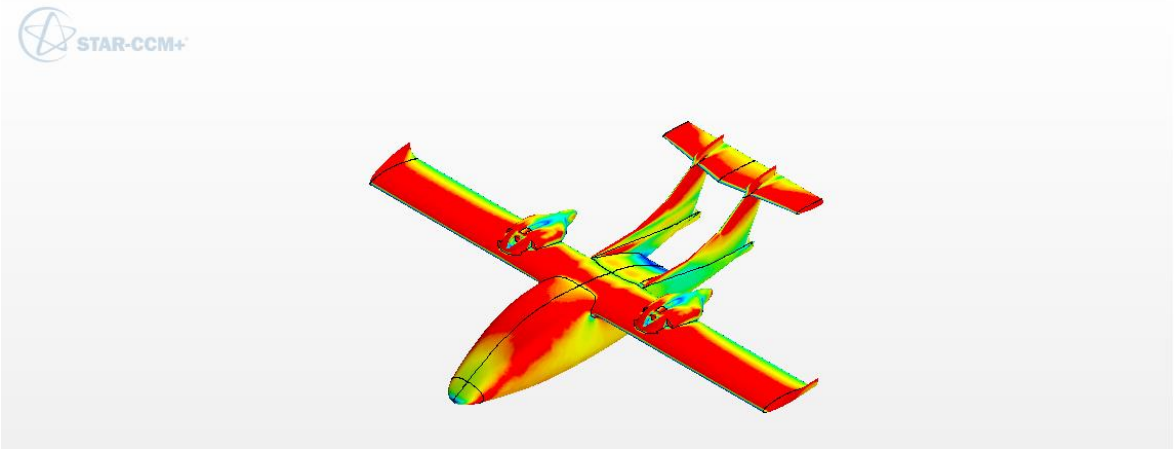


Figura 6.30: Cp caso 2


STAR-CCM+


STAR-CCM+


STAR-CCM+

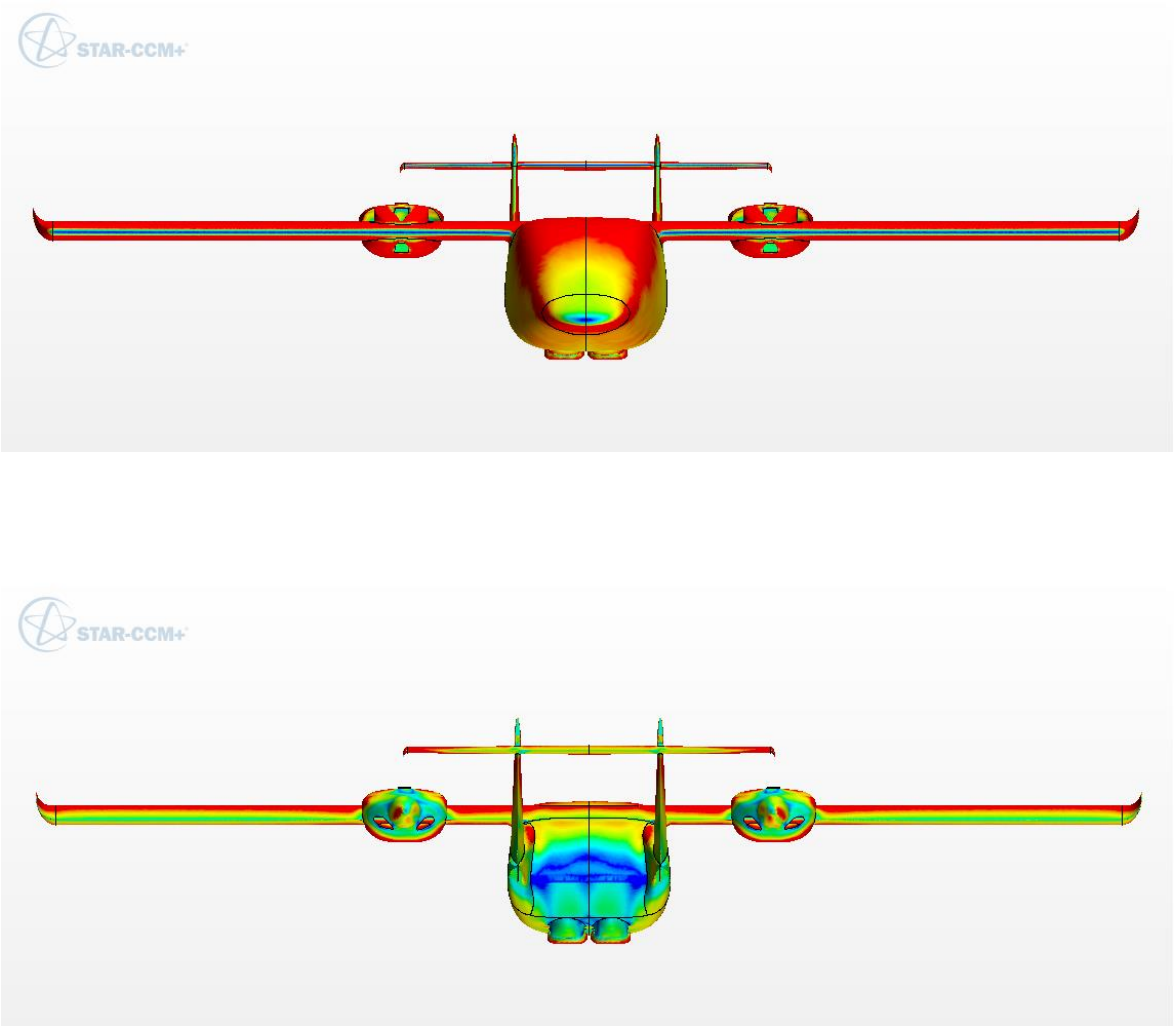


Figura 6.31: Wall Shear Stress caso 2

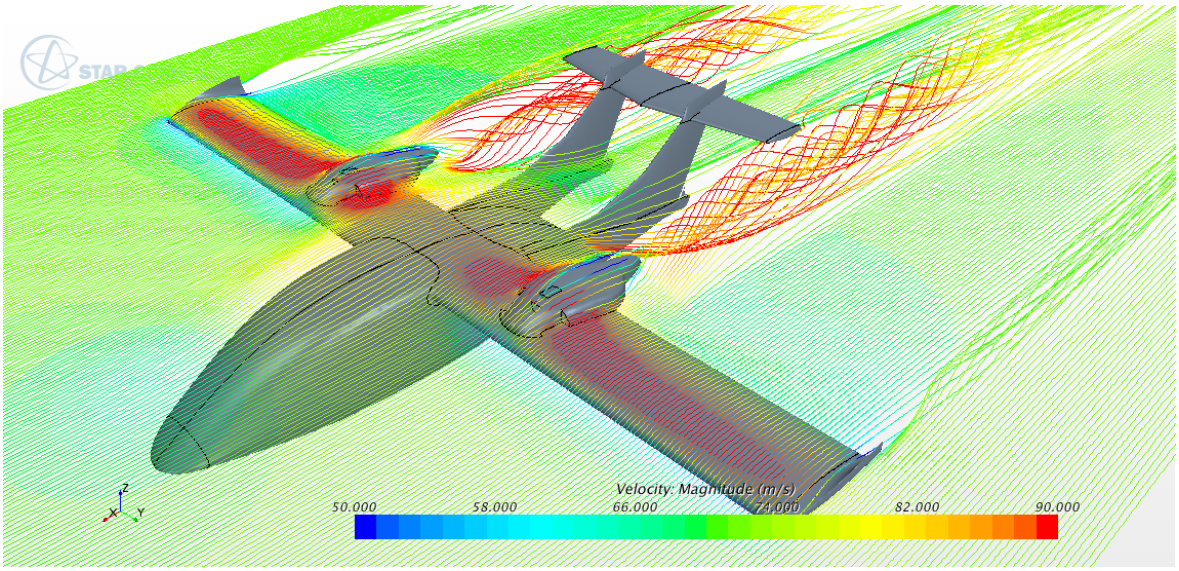


Figura 6.32: Linee di corrente caso 2

6.7 Caso 3

Infine nel caso 3 è stato simulato l'intero velivolo sia con la nuova configurazione del motore sia con i carrelli retraibili fin dentro la fusoliera.

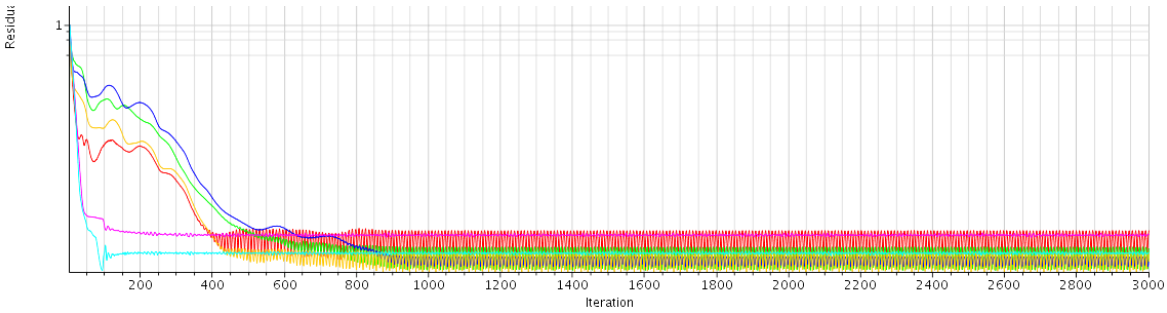


Figura 6.33: Residui caso 3

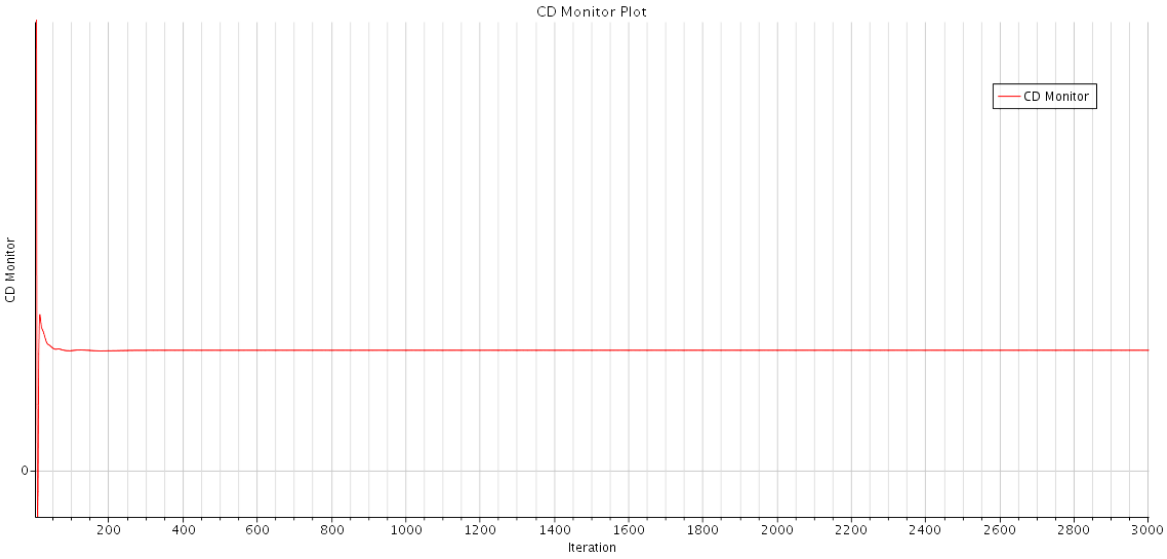


Figura 6.34: Convergenza Cd caso 3

Di seguito viene riportato il confronto con il caso iniziale e il conseguente guadagno in termini di resistenza percentuale:

	Δ (N)
Muso_DX	-1,57
Muso_SX	-1,55
Fusoliera_ant_DX	1,10
Fusoliera_ant_SX	1,30
Fusoliera_post_DX	1,08
Fusoliera_post_SX	-0,53
Coda_DX	-2,94
Coda_SX	-2,75
semiala_DX	-6,10
semiala_SX	-6,31
Winglet_DX	9,96
Winglet_SX	10,11
Motore_DX	-195,25
Motore_SX	-193,08
Impennaggio_oriz_DX	0,99
Impennaggio_oriz_SX	1,21
Winglet_coda_DX	-4,68
Winglet_coda_SX	-4,02
Pinna_alta_DX	-0,91
Pinna_alta_SX	-0,66
Pinna_bassa_DX	0,05
Pinna_bassa_SX	-0,02
Impennaggio_vert_DX	7,59
Impennaggio_vert_SX	8,70

Tabella 12: Differenze resistenza per PID caso 3

Anche qui, a parte la sezione dei motori, i valori sono pressoché uguali a quelli del caso iniziale.

	%
Fusoliera	9,3%
ALA	0,0%
Motori	16,0%
HTU	0,5%
VTU	-0,6%

Tabella 13: Guadagno percentuale resistenza caso 3

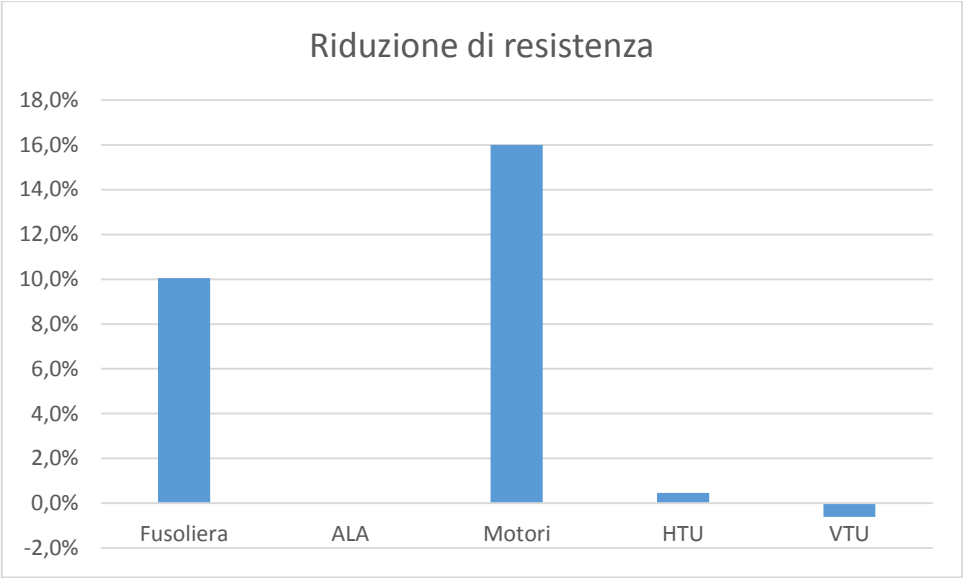
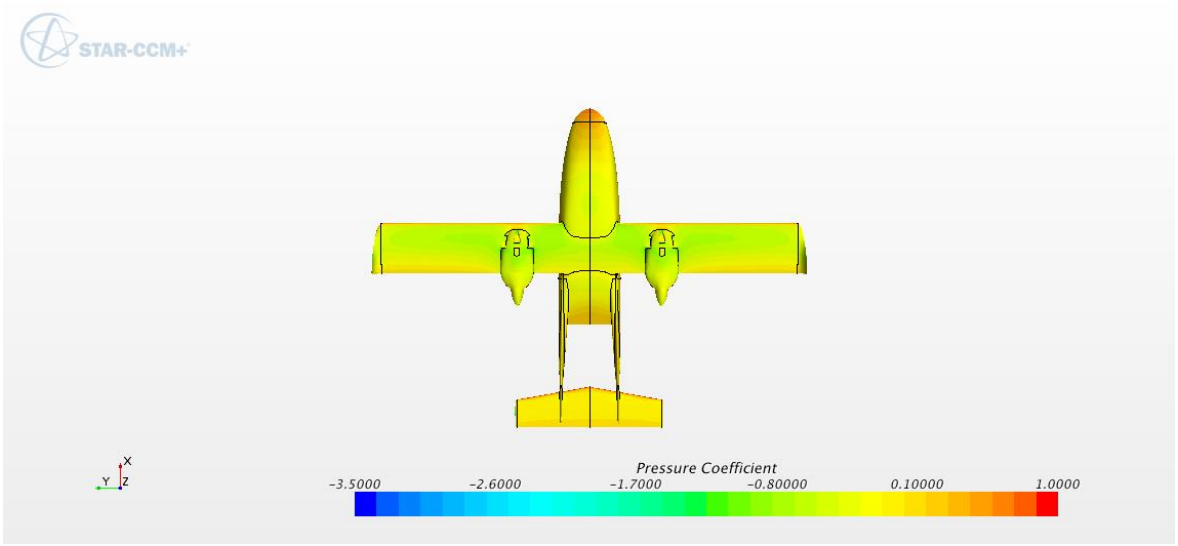
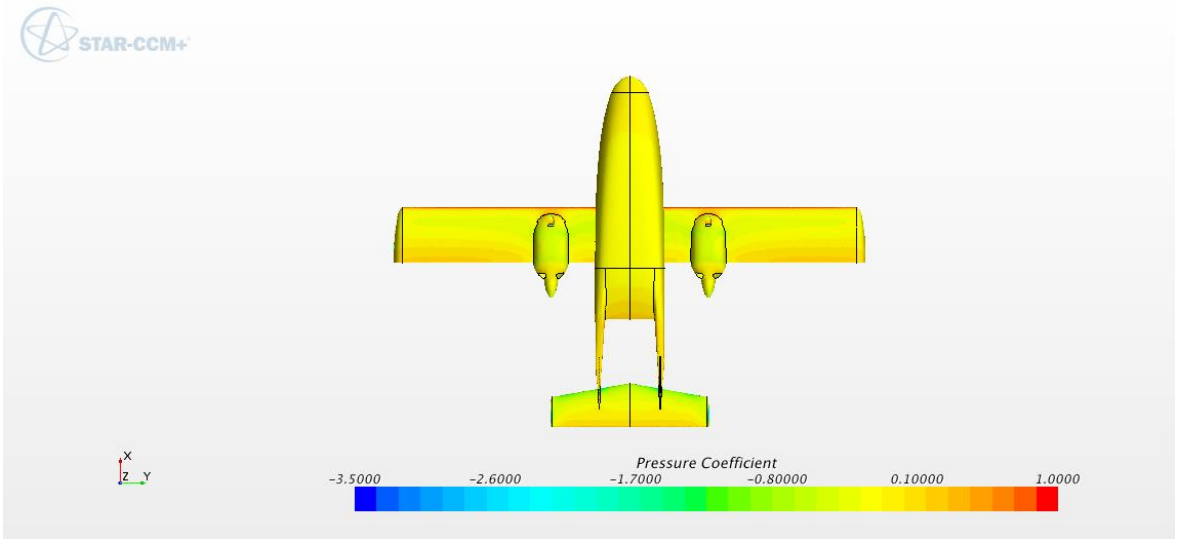
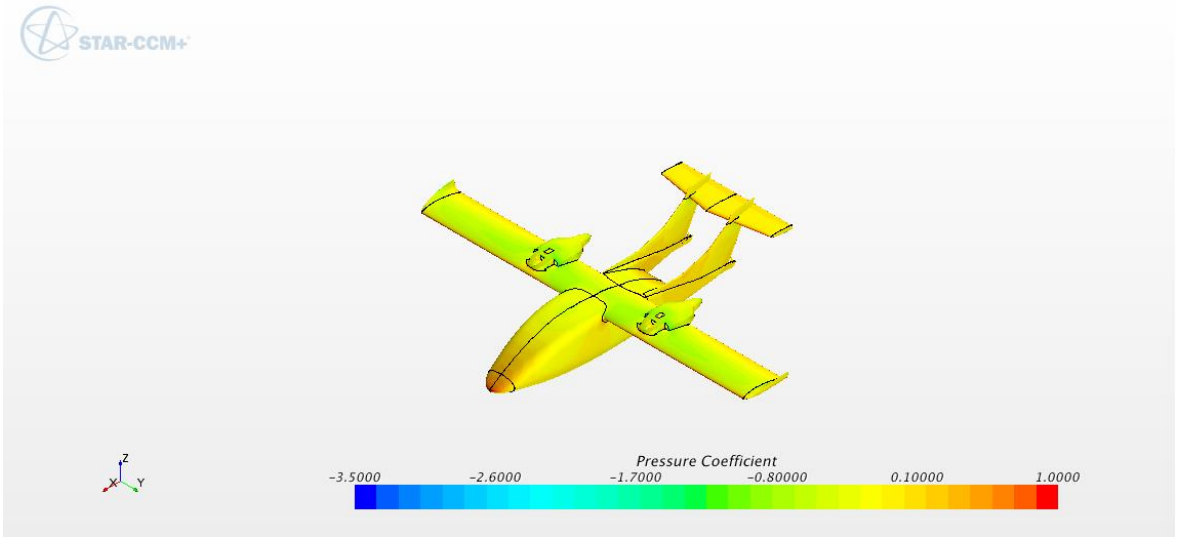


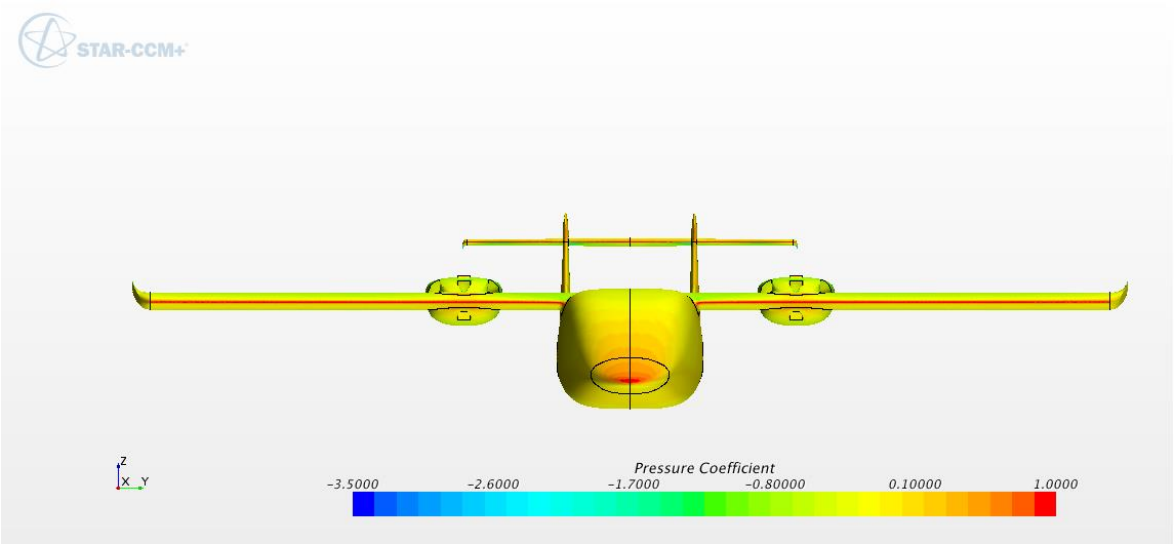
Figura 6.35: Istogramma riduzione di resistenza caso 3

La riduzione di resistenza ottenuta per il caso 3 è del 25,2% rispetto al caso iniziale. Verranno ora riportate le immagini dei coefficienti di pressione, degli sforzi tangenziali del campo di vorticità e la visualizzazione delle linee di corrente:


STAR-CCM+
X
Y
Z
Pressure Coefficient
-3.5000
-2.6000
-1.7000
-0.80000
0.10000
1.0000


STAR-CCM+
X
Z
Y
Pressure Coefficient
-3.5000
-2.6000
-1.7000
-0.80000
0.10000
1.0000


STAR-CCM+
Z
X
Y
Pressure Coefficient
-3.5000
-2.6000
-1.7000
-0.80000
0.10000
1.0000



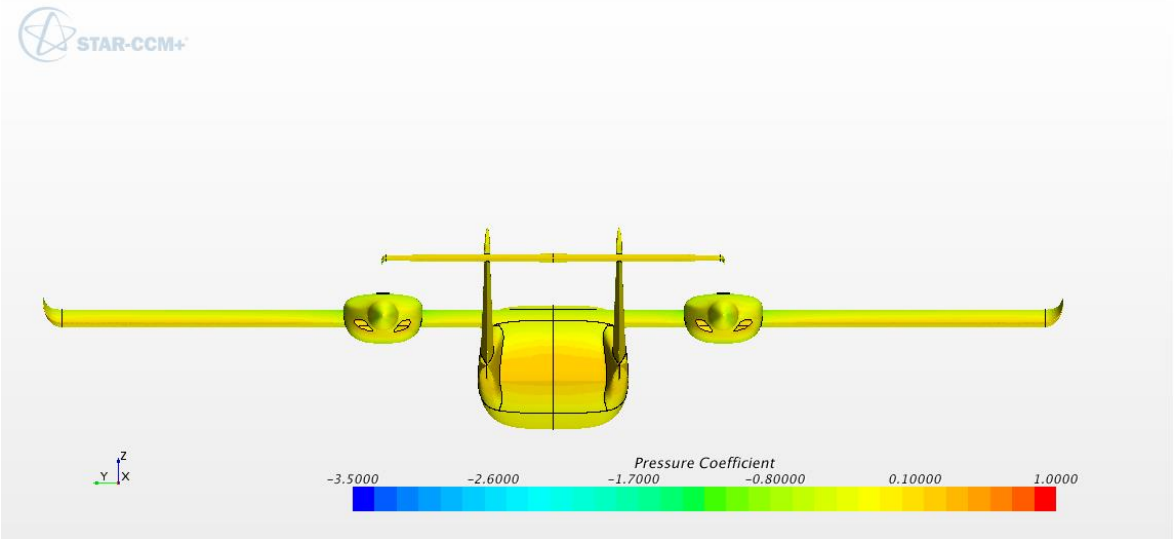
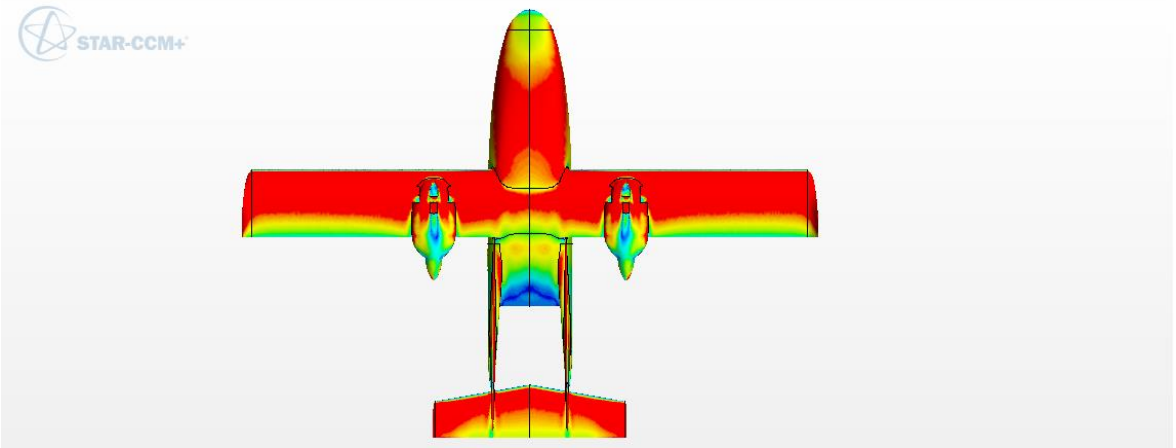


Figura 6.36: Cp caso 3



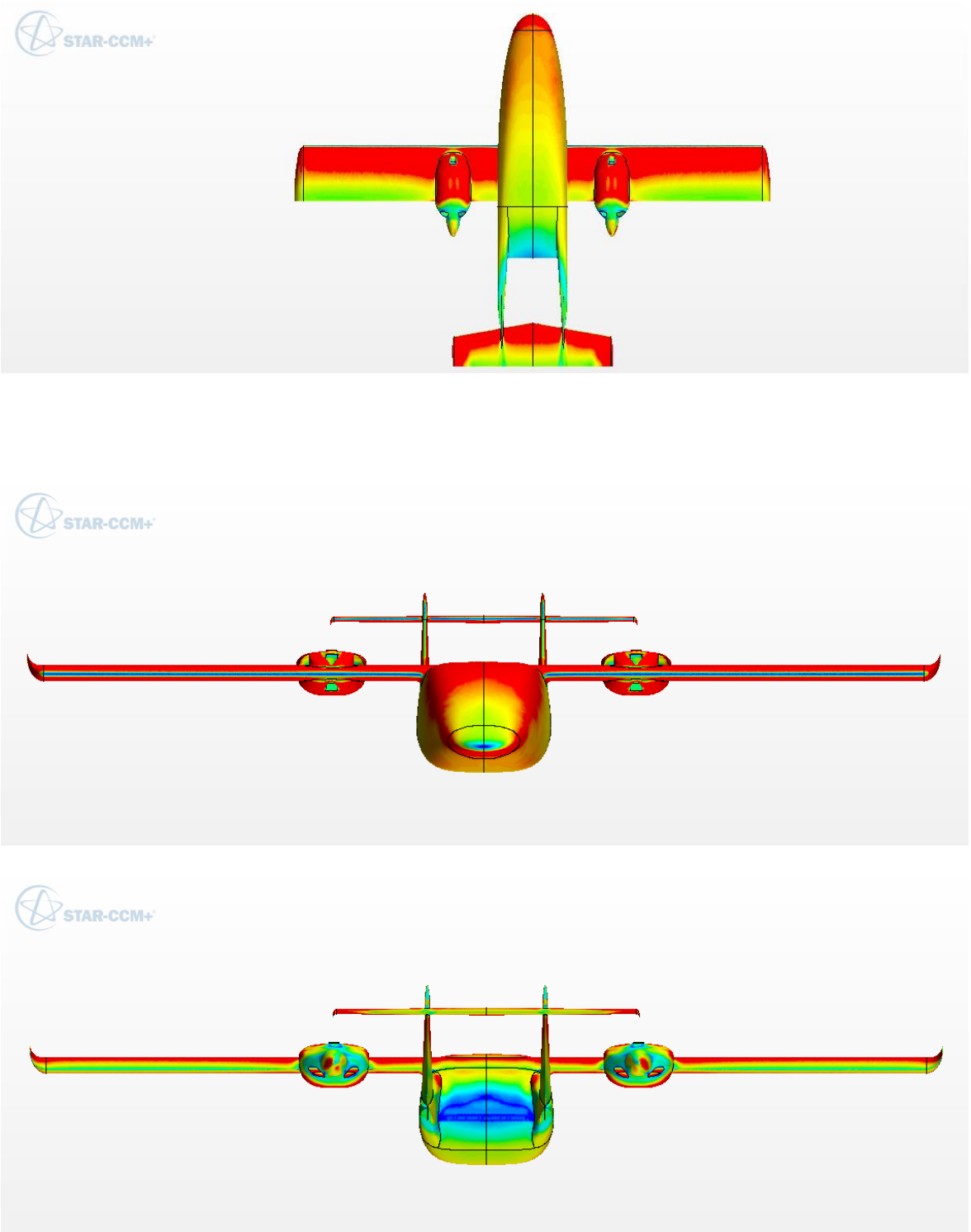
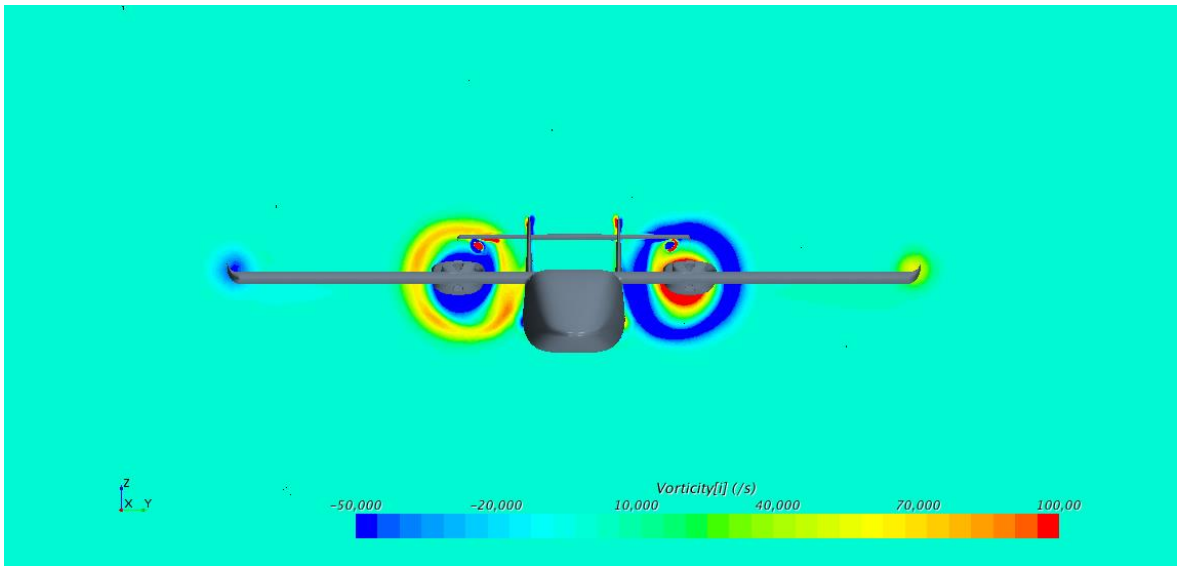
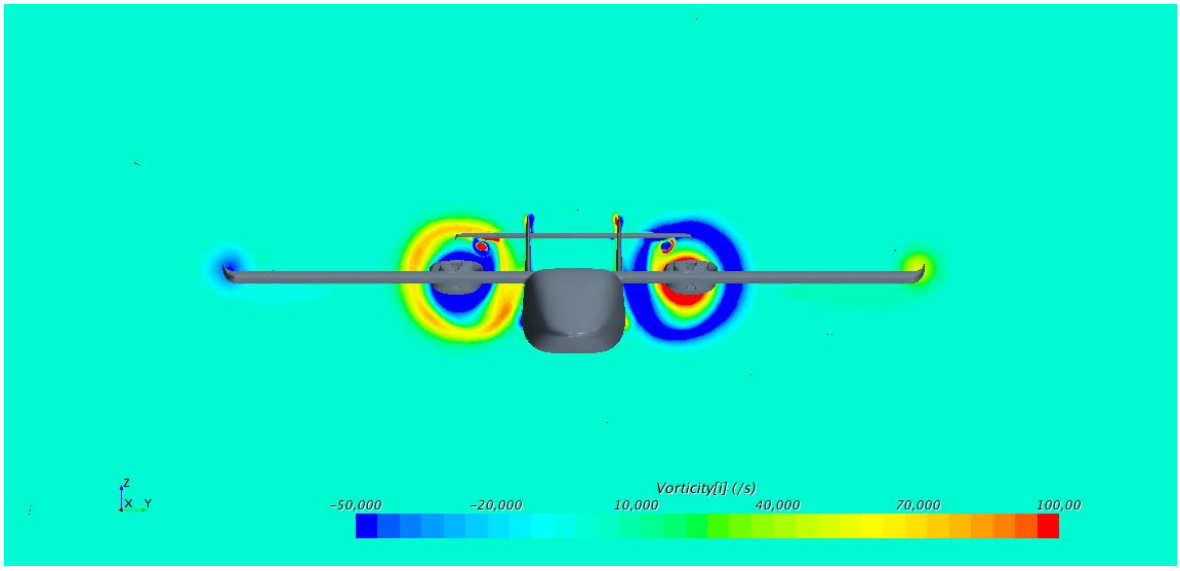
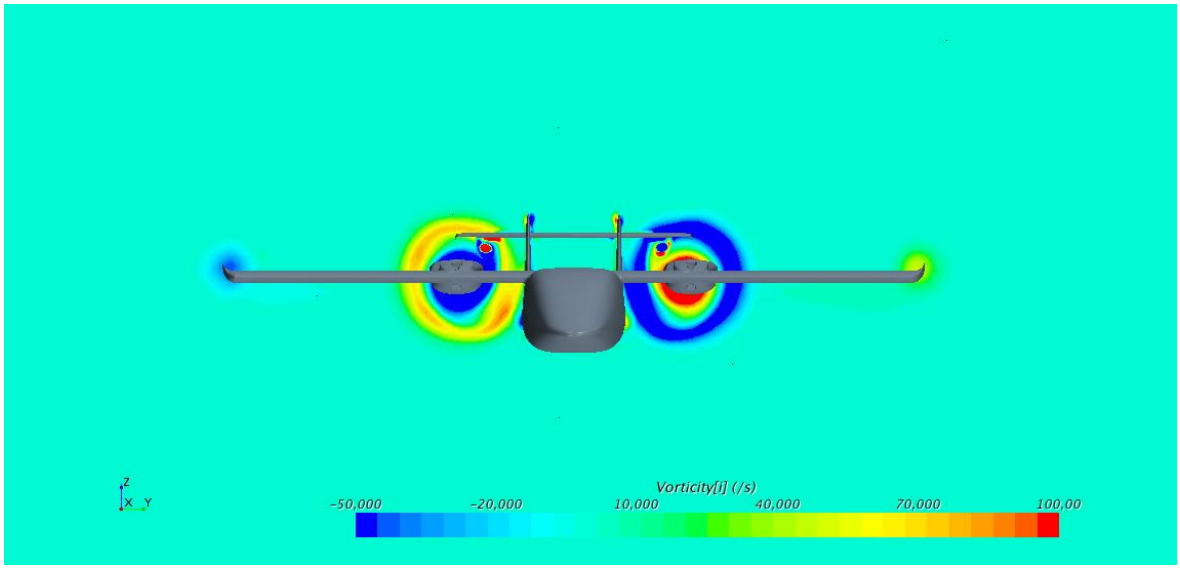


Figura 6.37: Wall Shear Stress caso 3


Z
X Y
Vorticity[i] (/s)
-50,000
-20,000
10,000
40,000
70,000
100,00


Z
X Y
Vorticity[i] (/s)
-50,000
-20,000
10,000
40,000
70,000
100,00


Z
X Y
Vorticity[i] (/s)
-50,000
-20,000
10,000
40,000
70,000
100,00

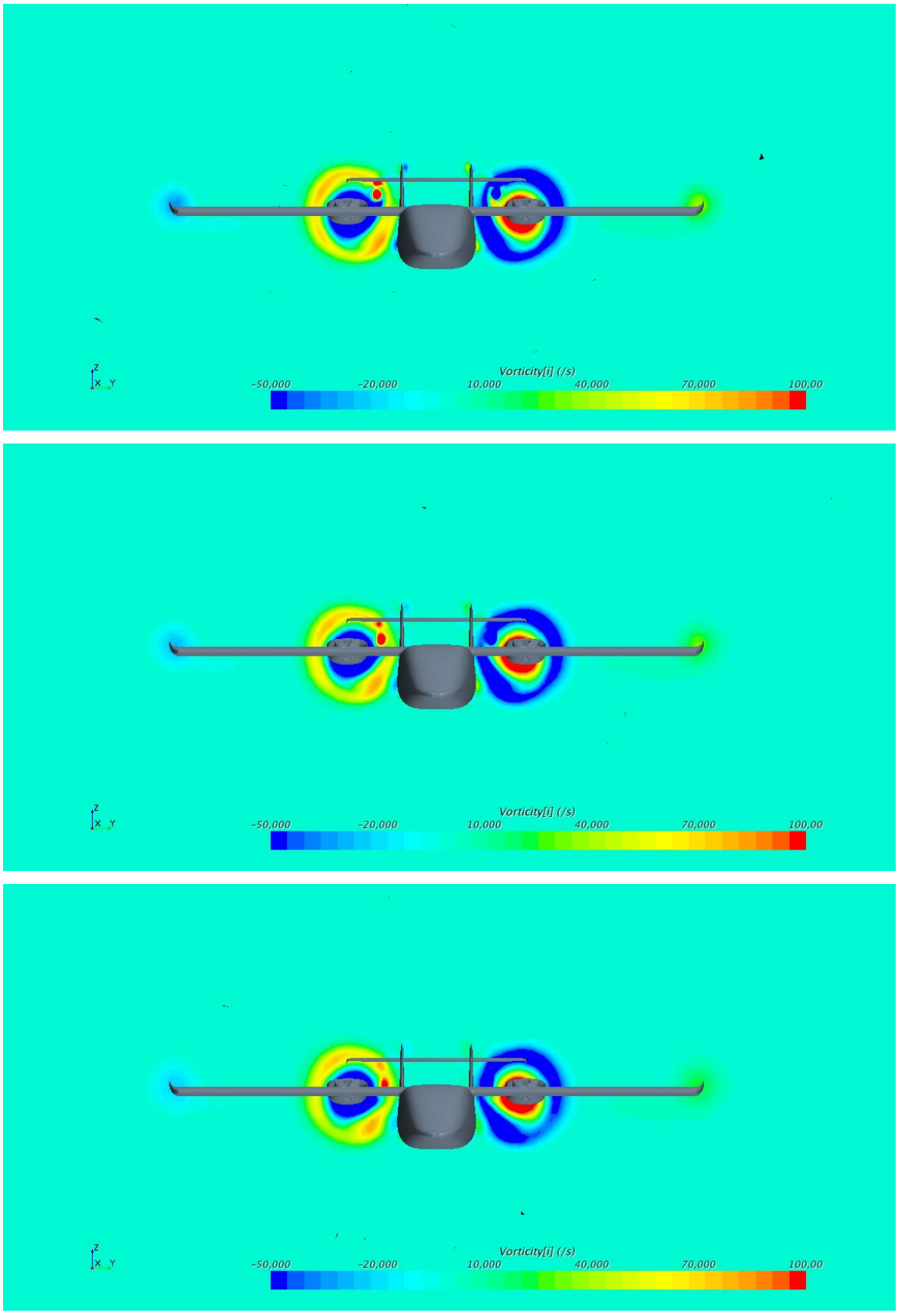


Figura 6.38: Vortici caso 3

Dall'alto in basso delle Figura 6.38, il piano trasversale, su cui è stata ottenuta la visualizzazione dei vortici, è rispettivamente a [0.2 , 0.4 , 0.6 , 1 , 1.5 , 2] m dal piano di coda.

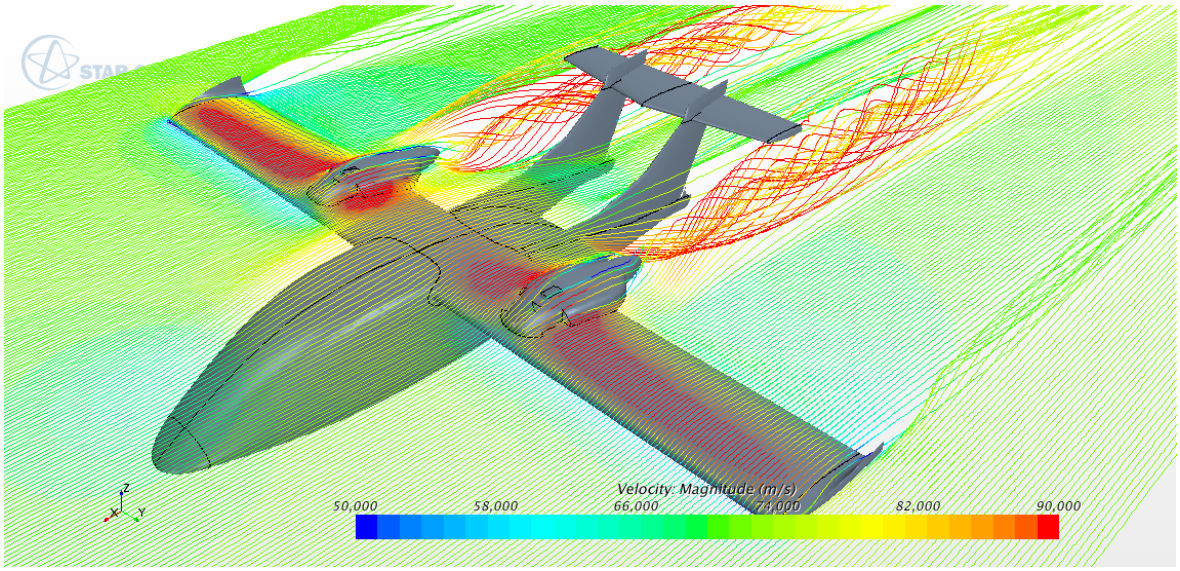


Figura 6.39: Linee di corrente caso 3

7 Conclusioni

Nel presente lavoro è stata realizzato un primo tentativo di modifica della geometria di un velivolo allo scopo di ridurne la resistenza. Sono stati utilizzati i software ANSA Beta CAE, per la mesh di superficie e Star CCM+ come solutore fluidodinamico.

La prima parte del lavoro è stato fare un'analisi CFD del velivolo per vedere come si distribuiscono le forze di resistenza e individuare opportunamente le componenti da modificare. Si sono scelti il reparto carrelli e le gondole motore, poiché assieme, generavano il 42% della resistenza totale del velivolo. Per i carrelli si è scelta una soluzione retraibile fin dentro la fusoliera. Per il motore, invece, è stata fatta dapprima un'analisi più dettagliata, suddividendo la carenatura del motore in più parti, così da comprendere effettivamente quale fosse quella critica. La parte in questione si è rivelata essere la parte posteriore della carenatura, che è stata modificata con una soluzione *boat-tail*.

Sono state quindi effettuate le analisi CFD di 3 casi:

1. Velivolo completo con gondola motore originale e senza carrelli
2. Velivolo completo con gondola motore nuovo e carrelli
3. Velivolo completo con gondola motore nuovo e senza carrelli

I guadagni in termini di resistenza sono stati valutati singolarmente per ogni caso. Nel caso 3 si è ottenuta una riduzione della resistenza pari al 25% (rispetto al caso di riferimento).

8 APPENDICE: Macro di Star CCM+

Si riporta la *macro*, scritta in linguaggio *.java*, utilizzata per l'automatizzazione della mesh di volume e del *run*.

```
import java.util.*;
import star.common.*;
import star.base.neo.*;
import star.vis.*;
import star.dualmesher.*;
import star.prismmesher.*;
import star.meshing.*;

public class Mesh extends StarMacro {

  public void execute() {
    execute0();
  }

  private void execute0() {

    Simulation simulation_0 =
      getActiveSimulation();

    Units units_0 =
      simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));

    ImportManager importManager_0 =
      simulation_0.getImportManager();

    Units units_1 =
      ((Units) simulation_0.getUnitsManager().getObject("mm"));

// IMPORTO GEOMETRIA

    importManager_0.importProstarSurface(resolvePath("Skycar.inp"),
"OneBoundaryPerPatch", true, units_1);

    SurfaceRep surfaceRep_0 =
      ((SurfaceRep) simulation_0.getRepresentationManager().getObject("Import"));

    Region region_0 =
      simulation_0.getRegionManager().getRegion("Region 1");

    surfaceRep_0.generateMeshReport(new NeoObjectVector(new Object[] {region_0}));
```

```
    MeshContinuum meshContinuum_0 =
      ((MeshContinuum) simulation_0.getContinuumManager().getContinuum("Mesh 1"));

    meshContinuum_0.getReferenceValues().get(BaseSize.class).setValue(2.5);

    meshContinuum_0.enable(DualMesherModel.class);

    meshContinuum_0.enable(PrismMesherModel.class);

// SETTAGGI MESH

    NumPrismLayers numPrismLayers_0 =
      meshContinuum_0.getReferenceValues().get(NumPrismLayers.class);

    numPrismLayers_0.setNumLayers(7);

    PrismLayerStretching prismLayerStretching_0 =
      meshContinuum_0.getReferenceValues().get(PrismLayerStretching.class);

    prismLayerStretching_0.setStretching(1.1);

    PrismThickness prismThickness_0 =
      meshContinuum_0.getReferenceValues().get(PrismThickness.class);


prismThickness_0.getRelativeOrAbsoluteOption().setSelected(RelativeOrAbsoluteOption.AB
SOLUTE);

    GenericAbsoluteSize genericAbsoluteSize_0 =
      ((GenericAbsoluteSize) prismThickness_0.getAbsoluteSize());

    genericAbsoluteSize_0.getValue().setValue(0.03);

    VolumeMeshDensity volumeMeshDensity_0 =
      meshContinuum_0.getReferenceValues().get(VolumeMeshDensity.class);

    volumeMeshDensity_0.setGrowthFactor(1.1);

// GENERAZIONE ZONA DI INFITTIMENTO

    MeshPartFactory meshPartFactory_0 =
      simulation_0.get(MeshPartFactory.class);

    SimpleCylinderPart simpleCylinderPart_0 =

meshPartFactory_0.createNewCylinderPart(simulation_0.get(SimulationPartManager.class));

    LabCoordinateSystem labCoordinateSystem_0 =
      simulation_0.getCoordinateSystemManager().getLabCoordinateSystem();

    simpleCylinderPart_0.setCoordinateSystem(labCoordinateSystem_0);
```

```
    simpleCylinderPart_0.getRadius().setUnits(units_0);

    simpleCylinderPart_0.getEndRadius().setUnits(units_0);

    Coordinate coordinate_4 =
      simpleCylinderPart_0.getStartCoordinate();

    coordinate_4.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {0.0,
0.0, 1.0}));

    Coordinate coordinate_5 =
      simpleCylinderPart_0.getEndCoordinate();

    coordinate_5.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {0.0,
0.0, 0.0}));

    simpleCylinderPart_0.getRadius().setValue(1.0);

    simpleCylinderPart_0.getEndRadius().setValue(1.0);

    coordinate_4.setValue(new DoubleVector(new double[] {-5.1, -2.0, 2.965}));

    coordinate_5.setValue(new DoubleVector(new double[] {-10.1, -2.0, 2.965}));


simpleCylinderPart_0.getTessellationDensityOption().setSelected(TessellationDensityOption.
MEDIUM);




    SimpleCylinderPart simpleCylinderPart_1 =

meshPartFactory_0.createNewCylinderPart(simulation_0.get(SimulationPartManager.class));

    simpleCylinderPart_1.setCoordinateSystem(labCoordinateSystem_0);

    simpleCylinderPart_1.getRadius().setUnits(units_0);

    simpleCylinderPart_1.getEndRadius().setUnits(units_0);

    Coordinate coordinate_6 =
      simpleCylinderPart_1.getStartCoordinate();

    coordinate_6.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {0.0,
0.0, 1.0}));

    Coordinate coordinate_7 =
      simpleCylinderPart_1.getEndCoordinate();
```

```
    coordinate_7.setCoordinate(units_0, units_0, units_0, new DoubleVector(new double[] {0.0,
0.0, 0.0}));

    simpleCylinderPart_1.getRadius().setValue(1.0);

    simpleCylinderPart_1.getEndRadius().setValue(1.0);

    coordinate_6.setValue(new DoubleVector(new double[] {-5.1, 2.0, 2.965}));

    coordinate_7.setValue(new DoubleVector(new double[] {-10.1, 2.0, 2.965}));


simpleCylinderPart_1.getTessellationDensityOption().setSelected(TessellationDensityOption.
MEDIUM);



    VolumeSource volumeSource_0 =
      meshContinuum_0.getVolumeSources().createVolumeSource();

    volumeSource_0.getPartGroup().setObjects(simpleCylinderPart_0);

    VolumeSourceDualMesherSizeOption volumeSourceDualMesherSizeOption_0 =

volumeSource_0.get(MeshConditionManager.class).get(VolumeSourceDualMesherSizeOptio
n.class);

    volumeSourceDualMesherSizeOption_0.setVolumeSourceDualMesherSizeOption(true);

    VolumeSourceSize volumeSourceSize_0 =
      volumeSource_0.get(MeshValueManager.class).get(VolumeSourceSize.class);


volumeSourceSize_0.getRelativeOrAbsoluteOption().setSelected(RelativeOrAbsoluteOption.
ABSOLUTE);

    GenericAbsoluteSize genericAbsoluteSize_1 =
      ((GenericAbsoluteSize) volumeSourceSize_0.getAbsoluteSize());

    genericAbsoluteSize_1.getValue().setValue(0.1);

    VolumeSource volumeSource_1 =
      meshContinuum_0.getVolumeSources().createVolumeSource();

    volumeSource_1.getPartGroup().setObjects(simpleCylinderPart_1);

    VolumeSourceDualMesherSizeOption volumeSourceDualMesherSizeOption_1 =

volumeSource_1.get(MeshConditionManager.class).get(VolumeSourceDualMesherSizeOptio
n.class);
```

```
    volumeSourceDualMesherSizeOption_1.setVolumeSourceDualMesherSizeOption(true);

    VolumeSourceSize volumeSourceSize_1 =
      volumeSource_1.get(MeshValueManager.class).get(VolumeSourceSize.class);


volumeSourceSize_1.getRelativeOrAbsoluteOption().setSelected(RelativeOrAbsoluteOption.
ABSOLUTE);

    GenericAbsoluteSize genericAbsoluteSize_2 =
      ((GenericAbsoluteSize) volumeSourceSize_1.getAbsoluteSize());

    genericAbsoluteSize_2.getValue().setValue(0.1);

    Boundary boundary_1 =
      region_0.getBoundaryManager().getBoundary("Inlet");


boundary_1.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_2 =
      region_0.getBoundaryManager().getBoundary("Inlet_Motore_DX");


boundary_2.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_3 =
      region_0.getBoundaryManager().getBoundary("Inlet_Motore_SX");


boundary_3.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_4 =
      region_0.getBoundaryManager().getBoundary("Outlet");


boundary_4.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_5 =
      region_0.getBoundaryManager().getBoundary("Outlet_up_Motore_SX");


boundary_5.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_6 =
```

```
      region_0.getBoundaryManager().getBoundary("Outlet_up_Motore_DX");


boundary_6.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_7 =
      region_0.getBoundaryManager().getBoundary("Outlet_Motore_SX");


boundary_7.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_8 =
      region_0.getBoundaryManager().getBoundary("Outlet_Motore_DX");


boundary_8.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    Boundary boundary_9 =
      region_0.getBoundaryManager().getBoundary("Symmetry");


boundary_9.get(MeshConditionManager.class).get(CustomizeBoundaryPrismsOption.class).s
etSelected(CustomizeBoundaryPrismsOption.DISABLE);

    MeshPipelineController meshPipelineController_0 =
      simulation_0.get(MeshPipelineController.class);

    meshPipelineController_0.generateVolumeMesh();
  }
}
```

E ora il *run* vero e proprio

```
import java.util.*;
import star.turbulence.*;
import star.material.*;
import star.common.*;
import star.base.neo.*;
import star.vis.*;
import star.flow.*;
```

```
import star.metrics.*;
import star.resurfacer.*;
import star.vis.*;
import star.meshing.*;
import star.vdm.*;
import star.dualmesher.*;
import star.prismmesher.*;
import star.keturb.*;
import star.segregatedflow.*;
import star.base.report.*;

public class Run extends StarMacro {

  public void execute() {
    execute0();
  }

  private void execute0() {

    Simulation simulation_0 =
      getActiveSimulation();

    Units units_0 =
      simulation_0.getUnitsManager().getPreferredUnits(new IntVector(new int[] {0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}));



PhysicsContinuum physicsContinuum_0 =
      simulation_0.getContinuumManager().createContinuum(PhysicsContinuum.class);

// SETTAGGI MODELLO FISICO

    physicsContinuum_0.enable(ThreeDimensionalModel.class);

    physicsContinuum_0.enable(SteadyModel.class);

    physicsContinuum_0.enable(SingleComponentGasModel.class);

    physicsContinuum_0.enable(SegregatedFlowModel.class);

    physicsContinuum_0.enable(ConstantDensityModel.class);

    physicsContinuum_0.enable(TurbulentModel.class);

    physicsContinuum_0.enable(RansTurbulenceModel.class);

    physicsContinuum_0.enable(KEpsilonTurbulence.class);

    physicsContinuum_0.enable(RkeTwoLayerTurbModel.class);
```

```
    physicsContinuum_0.enable(KeTwoLayerAllYplusWallTreatment.class);

    physicsContinuum_0.enable(VirtualDiskModel.class);

    physicsContinuum_0.enable(CellQualityRemediationModel.class);

// IMPORTA DATI DELL'ELICA

    FileTable fileTable_0 =
      (FileTable) simulation_0.getTableManager().createFromFile(resolvePath("Cartel1.csv"));

//VIRTUAL DISK MODEL

    VirtualDiskModel virtualDiskModel_0 =
      physicsContinuum_0.getModelManager().getModel(VirtualDiskModel.class);

    VirtualDisk virtualDisk_0 =
      virtualDiskModel_0.getVirtualDiskManager().createDisk("Virtual Disk");

    virtualDisk_0.setActiveMethod(BodyForcePropellerMethod.class);


virtualDisk_0.getComponentsManager().get(PropellerCurve.class).setActiveMethod(Propelle
rCurveTableMethod.class);

    PropellerCurveTableMethod propellerCurveTableMethod_0 =
      ((PropellerCurveTableMethod)
virtualDisk_0.getComponentsManager().get(PropellerCurve.class).getActiveMethod());

    propellerCurveTableMethod_0.setTable(fileTable_0);

    propellerCurveTableMethod_0.setAdvanceRatio("J");

    propellerCurveTableMethod_0.setKt("Kt");

    propellerCurveTableMethod_0.setKq("Kq");

    propellerCurveTableMethod_0.setEta("Eta");

    SimpleDiskGeometry simpleDiskGeometry_0 =
      virtualDisk_0.getComponentsManager().get(SimpleDiskGeometry.class);

    ((NormalAndCoordinateSystem)
simpleDiskGeometry_0.getOrientationSpecification()).getDiskNormal().setComponents(1.0,
0.0, 0.0);

    Coordinate coordinate_0 =
      simpleDiskGeometry_0.getDiskOrigin();

    Units units_2 =
      ((Units) simulation_0.getUnitsManager().getObject("m"));
```

```
    coordinate_0.setCoordinate(units_2, units_2, units_2, new DoubleVector(new double[] {-
7.6, -2.0, 2.965}));

    PropellerInflowVelocityPlane propellerInflowVelocityPlane_0 =
      virtualDisk_0.getComponentsManager().get(PropellerInflowVelocityPlane.class);

    propellerInflowVelocityPlane_0.getRadius().setValue(1.1);

    propellerInflowVelocityPlane_0.getOffset().setValue(0.1);

    PropellerRotationRateInputValue propellerRotationRateInputValue_0 =
      virtualDisk_0.getComponentsManager().get(PropellerRotationRateInputValue.class);

    propellerRotationRateInputValue_0.getRotationRate().setValue(2450.0);

    propellerRotationRateInputValue_0.getRotationRate().setValue(2450.0);

    simpleDiskGeometry_0.getDiskInnerRadius().setValue(0.15);




    VirtualDisk virtualDisk_1 =
      virtualDiskModel_0.getVirtualDiskManager().createDisk("Virtual Disk");

    virtualDisk_1.setActiveMethod(BodyForcePropellerMethod.class);


virtualDisk_1.getComponentsManager().get(PropellerCurve.class).setActiveMethod(Propelle
rCurveTableMethod.class);

    PropellerCurveTableMethod propellerCurveTableMethod_1 =
      ((PropellerCurveTableMethod)
virtualDisk_1.getComponentsManager().get(PropellerCurve.class).getActiveMethod());

    propellerCurveTableMethod_1.setTable(fileTable_0);

    propellerCurveTableMethod_1.setAdvanceRatio("J");

    propellerCurveTableMethod_1.setKt("Kt");

    propellerCurveTableMethod_1.setKq("Kq");

    propellerCurveTableMethod_1.setEta("Eta");

    SimpleDiskGeometry simpleDiskGeometry_1 =
      virtualDisk_1.getComponentsManager().get(SimpleDiskGeometry.class);
```

```
   ((NormalAndCoordinateSystem)
simpleDiskGeometry_1.getOrientationSpecification()).getDiskNormal().setComponents(1.0,
0.0, 0.0);

   Coordinate coordinate_1 =
     simpleDiskGeometry_1.getDiskOrigin();

   coordinate_1.setCoordinate(units_2, units_2, units_2, new DoubleVector(new double[] {-
7.6, 2.0, 2.965}));

   PropellerInflowVelocityPlane propellerInflowVelocityPlane_1 =
     virtualDisk_1.getComponentsManager().get(PropellerInflowVelocityPlane.class);

   propellerInflowVelocityPlane_1.getRadius().setValue(1.1);

   propellerInflowVelocityPlane_1.getOffset().setValue(0.1);

   PropellerRotationRateInputValue propellerRotationRateInputValue_1 =
     virtualDisk_1.getComponentsManager().get(PropellerRotationRateInputValue.class);

   propellerRotationRateInputValue_1.getRotationRate().setValue(2450.0);

   propellerRotationRateInputValue_1.getRotationRate().setValue(2450.0);

   simpleDiskGeometry_1.getDiskInnerRadius().setValue(0.15);



   Region region_0 =
     simulation_0.getRegionManager().getRegion("Region 1");

   Boundary boundary_0 =
     region_0.getBoundaryManager().getBoundary("semiala_SX");

   Boundary boundary_1 =
     region_0.getBoundaryManager().getBoundary("semiala_DX");

   Boundary boundary_2 =
     region_0.getBoundaryManager().getBoundary("Winglet_SX");

   Boundary boundary_3 =
     region_0.getBoundaryManager().getBoundary("Winglet_DX");

   Boundary boundary_4 =
     region_0.getBoundaryManager().getBoundary("Fusoliera_ant_DX");

   Boundary boundary_5 =
     region_0.getBoundaryManager().getBoundary("Fusoliera_ant_SX");

   Boundary boundary_6 =
     region_0.getBoundaryManager().getBoundary("Muso_DX");
```

```
Boundary boundary_7 =
  region_0.getBoundaryManager().getBoundary("Muso_SX");

Boundary boundary_8 =
  region_0.getBoundaryManager().getBoundary("Fusoliera_post_DX");

Boundary boundary_9 =
  region_0.getBoundaryManager().getBoundary("Fusoliera_post_SX");

Boundary boundary_10 =
  region_0.getBoundaryManager().getBoundary("Coda_DX");

Boundary boundary_11 =
  region_0.getBoundaryManager().getBoundary("Pinna_bassa_DX");

Boundary boundary_12 =
  region_0.getBoundaryManager().getBoundary("Impennaggio_vert_DX");

Boundary boundary_13 =
  region_0.getBoundaryManager().getBoundary("Coda_SX");

Boundary boundary_14 =
  region_0.getBoundaryManager().getBoundary("Pinna_bassa_SX");

Boundary boundary_15 =
  region_0.getBoundaryManager().getBoundary("Impennaggio_vert_SX");

Boundary boundary_16 =
  region_0.getBoundaryManager().getBoundary("Impennaggio_oriz_SX");

Boundary boundary_17 =
  region_0.getBoundaryManager().getBoundary("Impennaggio_oriz_DX");

Boundary boundary_18 =
  region_0.getBoundaryManager().getBoundary("Winglet_coda_DX");

Boundary boundary_19 =
  region_0.getBoundaryManager().getBoundary("Winglet_coda_SX");

Boundary boundary_20 =
  region_0.getBoundaryManager().getBoundary("Motore_SX");

Boundary boundary_21 =
  region_0.getBoundaryManager().getBoundary("Motore_DX");

Boundary boundary_22 =
  region_0.getBoundaryManager().getBoundary("Carrello_SX");

Boundary boundary_23 =
  region_0.getBoundaryManager().getBoundary("Carrello_DX");
```

```
    Boundary boundary_24 =
      region_0.getBoundaryManager().getBoundary("Pinna_alta_DX");

    Boundary boundary_25 =
      region_0.getBoundaryManager().getBoundary("Pinna_alta_SX");

    Boundary boundary_26 =
      region_0.getBoundaryManager().getBoundary("Symmetry");

    Boundary boundary_27 =
      region_0.getBoundaryManager().getBoundary("Inlet");

    Boundary boundary_28 =
      region_0.getBoundaryManager().getBoundary("Outlet");

    Boundary boundary_29 =
      region_0.getBoundaryManager().getBoundary("Inlet_Motore_DX");

    Boundary boundary_30 =
      region_0.getBoundaryManager().getBoundary("Outlet_Motore_DX");

    Boundary boundary_31 =
      region_0.getBoundaryManager().getBoundary("Outlet_Motore_SX");

    Boundary boundary_32 =
      region_0.getBoundaryManager().getBoundary("Inlet_Motore_SX");

    Boundary boundary_33 =
      region_0.getBoundaryManager().getBoundary("Outlet_up_Motore_DX");

    Boundary boundary_34 =
      region_0.getBoundaryManager().getBoundary("Outlet_up_Motore_SX");

// CONDIZIONI INIZIALI E AL CONTORNO

    VelocityMagnitudeProfile velocityMagnitudeProfile_0 =
      boundary_27.getValues().get(VelocityMagnitudeProfile.class);


velocityMagnitudeProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().set
Value(72.0);


velocityMagnitudeProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().set
Value(72.0);

    boundary_28.setBoundaryType(PressureBoundary.class);

    boundary_29.setBoundaryType(OutletBoundary.class);
```

```
boundary_29.getConditions().get(OutflowSpecificationOption.class).setSelected(OutflowSpecificationOption.SPECIFIED_MASSFLUXES);

    MassFlowRateProfile massFlowRateProfile_0 =
      boundary_29.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(2.0);

    boundary_32.setBoundaryType(OutletBoundary.class);


boundary_32.getConditions().get(OutflowSpecificationOption.class).setSelected(OutflowSpecificationOption.SPECIFIED_MASSFLUXES);

    MassFlowRateProfile massFlowRateProfile_1 =
      boundary_32.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_1.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(2.0);


   boundary_33.setBoundaryType(MassFlowBoundary.class);

    MassFlowRateProfile massFlowRateProfile_2 =
      boundary_33.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_2.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(0.5);

  boundary_34.setBoundaryType(MassFlowBoundary.class);

    MassFlowRateProfile massFlowRateProfile_3 =
      boundary_34.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_3.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(0.5);

   boundary_31.setBoundaryType(MassFlowBoundary.class);

    MassFlowRateProfile massFlowRateProfile_4 =
      boundary_31.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_4.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValue(1.5);
```

```
    boundary_30.setBoundaryType(MassFlowBoundary.class);

    MassFlowRateProfile massFlowRateProfile_5 =
      boundary_30.getValues().get(MassFlowRateProfile.class);


massFlowRateProfile_5.getMethod(ConstantScalarProfileMethod.class).getQuantity().setVal
ue(1.5);

    InitialPressureProfile initialPressureProfile_0 =
      physicsContinuum_0.getInitialConditions().get(InitialPressureProfile.class);


initialPressureProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValu
e(0.0);


initialPressureProfile_0.getMethod(ConstantScalarProfileMethod.class).getQuantity().setValu
e(0.0);

    VelocityProfile velocityProfile_0 =
      physicsContinuum_0.getInitialConditions().get(VelocityProfile.class);


velocityProfile_0.getMethod(ConstantVectorProfileMethod.class).getQuantity().setCompone
nts(-72.0, 0.0, 0.0);


physicsContinuum_0.getReferenceValues().get(ReferencePressure.class).setValue(87514.0);

// STOPPING CRITERIA

    StepStoppingCriterion stepStoppingCriterion_0 =
      ((StepStoppingCriterion)
simulation_0.getSolverStoppingCriterionManager().getSolverStoppingCriterion("Maximum
Steps"));

    stepStoppingCriterion_0.setMaximumNumberSteps(3000);


    ForceCoefficientReport forceCoefficientReport_0 =
      simulation_0.getReportManager().createReport(ForceCoefficientReport.class);

    forceCoefficientReport_0.setPresentationName("CD");

    forceCoefficientReport_0.getDirection().setComponents(-1.0, 0.0, 0.0);

    forceCoefficientReport_0.getParts().setObjects(boundary_23, boundary_22, boundary_10,
boundary_13, boundary_4, boundary_5, boundary_8, boundary_9, boundary_17,
boundary_16, boundary_12, boundary_15, boundary_21, boundary_20, boundary_6,
```

```
boundary_7, boundary_24, boundary_25, boundary_11, boundary_14, boundary_1, boundary_0, boundary_18, boundary_19, boundary_3, boundary_2);

    forceCoefficientReport_0.getReferenceArea().setValue(16.8);

    forceCoefficientReport_0.getReferenceVelocity().setValue(72.0);

    forceCoefficientReport_0.getReferenceDensity().setValue(1.088);

    SurfaceRep surfaceRep_0 =
      ((SurfaceRep) simulation_0.getRepresentationManager().getObject("Import"));

    forceCoefficientReport_0.setRepresentation(surfaceRep_0);

    simulation_0.getMonitorManager().createMonitorAndPlot(new NeoObjectVector(new Object[] {forceCoefficientReport_0}), true, "%1$s Plot");

    ReportMonitor reportMonitor_0 =
      ((ReportMonitor) simulation_0.getMonitorManager().getMonitor("CD Monitor"));

    MonitorPlot monitorPlot_0 =
      simulation_0.getPlotManager().createMonitorPlot(new NeoObjectVector(new Object[] {reportMonitor_0}), "CD Monitor Plot");

    ResidualPlot residualPlot_0 =
      ((ResidualPlot) simulation_0.getPlotManager().getPlot("Residuals"));



ForceCoefficientReport forceCoefficientReport_1 =
      simulation_0.getReportManager().createReport(ForceCoefficientReport.class);

    forceCoefficientReport_1.getReferenceArea().setValue(16.8);

    forceCoefficientReport_1.getReferenceVelocity().setValue(72.0);

    forceCoefficientReport_1.getReferenceDensity().setValue(1.088);

forceCoefficientReport_1.getParts().setObjects(boundary_23, boundary_22, boundary_10, boundary_13, boundary_4, boundary_5, boundary_8, boundary_9, boundary_17, boundary_16, boundary_12, boundary_15, boundary_21, boundary_20, boundary_6, boundary_7, boundary_24, boundary_25, boundary_11, boundary_14, boundary_1, boundary_0, boundary_18, boundary_19, boundary_3, boundary_2);

   forceCoefficientReport_1.getDirection().setComponents(0.0, 0.0, 1.0);

    forceCoefficientReport_1.setPresentationName("CL");

    forceCoefficientReport_1.setRepresentation(surfaceRep_0);
```

```
   simulation_0.getMonitorManager().createMonitorAndPlot(new NeoObjectVector(new Object[] {forceCoefficientReport_1}), true, "%1$s Plot");

   ReportMonitor reportMonitor_1 =
     ((ReportMonitor) simulation_0.getMonitorManager().getMonitor("CL Monitor"));

   MonitorPlot monitorPlot_1 =
     simulation_0.getPlotManager().createMonitorPlot(new NeoObjectVector(new Object[] {reportMonitor_1}), "CL Monitor Plot");



   MomentCoefficientReport momentCoefficientReport_0 =
     simulation_0.getReportManager().createReport(MomentCoefficientReport.class);

   momentCoefficientReport_0.setRepresentation(surfaceRep_0);

   momentCoefficientReport_0.getReferenceDensity().setValue(1.088);

   momentCoefficientReport_0.getReferenceVelocity().setValue(72.0);

   momentCoefficientReport_0.getReferenceArea().setValue(16.8);

   momentCoefficientReport_0.getParts().setObjects(boundary_23, boundary_22, boundary_10, boundary_13, boundary_4, boundary_5, boundary_8, boundary_9, boundary_17, boundary_16, boundary_12, boundary_15, boundary_21, boundary_20, boundary_6, boundary_7, boundary_24, boundary_25, boundary_11, boundary_14, boundary_1, boundary_0, boundary_18, boundary_19, boundary_3, boundary_2);

momentCoefficientReport_0.getReferenceRadius().setValue(1.4);

momentCoefficientReport_0.setPresentationName("CM");

momentCoefficientReport_0.getDirection().setComponents(0.0, -1.0, 0.0);

momentCoefficientReport_0.getOrigin().setComponents(-5.917, 0.0, 2.904);

   simulation_0.getMonitorManager().createMonitorAndPlot(new NeoObjectVector(new Object[] {momentCoefficientReport_0}), true, "%1$s Plot");

   ReportMonitor reportMonitor_2 =
     ((ReportMonitor) simulation_0.getMonitorManager().getMonitor("CM Monitor"));

   MonitorPlot monitorPlot_2 =
     simulation_0.getPlotManager().createMonitorPlot(new NeoObjectVector(new Object[] {reportMonitor_2}), "CM Monitor Plot");

   residualPlot_0.open();

   monitorPlot_0.open();
```

```
    monitorPlot_1.open();

    monitorPlot_2.open();

    simulation_0.getSimulationIterator().run();

    }
}
```

9 BIBLIOGRAFIA

[1] BETA CAE-ANSA, *"Ansa_v15.0_users_guide"*

[2] CD-Adapco, *"User Guide - Star-CCM+ Version 8.04.007"*

[3] G. Lombardi, *"Dispense del corso di Aerodinamica degli Aeromobili"*, Università di Pisa, Dipartimento di Ingegneria Aerospaziale

[4] Pierre Sagaut, Sebastien Deck, Marc Terracol, *"Multiscale and multiresolution approaches in turbulence"*

[5] Egbert Torenbeek, *"Aerodynamic drag and its reduction"*

RINGRAZIAMENTI

Il primo ringraziamento va ai miei genitori, che non hanno mai smesso di credere in me nonostante tutti i guai che ho combinato. Un semplice grazie non basta mai; spero solo di essere in grado di ricambiare tutti i loro sacrifici un giorno e renderli fieri di me.

Grazie a mio fratello Salvo, che è sempre stato in grado, forse senza neanche rendersene conto, di farmi andare avanti nei momenti più difficili, riuscendo a strapparmi qualche sorriso.

Grazie a tutti quelli che mi hanno accompagnato in questi splendidi anni a Pisa. In particolar modo e in ordine assolutamente sparso:

grazie all'Avvocato per avermi fatto conoscere il mondo dei fumetti;

grazie a Jack per essere riuscito a sopportarmi fin dai tempi del liceo;

grazie a Giulia perché è l'unica con cui posso confidarmi davvero;

grazie a Giò, Cate, Peppa per le uscite del sabato sera;

grazie a Mariapaola per aver affinato il mio italiano (anche se, lo so, c'è ancora molto da fare);

grazie a quelli dell'aperitivo;

grazie a Maurilio per avermi fatto conoscere, anche se per poco, il mondo di D&D

grazie a Giulia (di Firenze) per avermi offerto la sua comprensione quando più ne avevo bisogno, nonostante non mi fossi fatto sentire per ben 9 anni;

grazie ai ragazzi della galleria: Marco ed Elena per l’aiuto, Gigi per i caffè (dai scherzo, in bocca al lupo per tutto!) e tutti gli altri per la compagnia;

Infine, grazie a chiunque altro abbia mai conosciuto e che non ho fin qui nominato (molto probabilmente perché ho già scordato il nome).